CSC-EPL-88/003

2

NATIONAL COMPUTER SECURITY CENTER

FINAL EVALUATION REPORT

OF

INTERNATIONAL BUSINESS MACHINES CORPORATION

MVS/XA with RACF Version 1.8

15 June 1988

Approved for Public Release:
Distribution Unlimited

FINAL EVALUATION REPORT

INTERNATIONAL BUSINESS MACHINES CORPORATION

MVS/XA with RACF

NATIONAL
COMPUTER SECURITY CENTER

9800 Savage Road
Fort George G. Meade
Maryland 20755-6000

15 June 1988

CSC-EPL-88/003
Library No. ~~S230,621~~
S230,620

This page intentionally left blank.

FOREWORD

This publication, the Final Evaluation Report International Business Machines Corporation, MVS/XA with RACF, is being issued by the National Computer Security Center under the authority of and in accordance with DoD Directive 5215.1, "Computer Security Evaluation Center". The purpose of this report is to document the results of the formal evaluation of IBM's MVS/XA with RACF operating system. The requirements stated in this report are taken from *DEPARTMENT OF DEFENSE TRUSTED COMPUTER SYSTEM EVALUATION CRITERIA* dated December 1985.

Approved:

Eliot Sohmer 15 June 1988

Eliot Sohmer
Chief, Office of Computer Security
Evaluations, Publications, and Support
National Computer Security Center

ACKNOWLEDGEMENTS

Team Members

Team members included the following individuals:

Jerzy W. Rub

Kenneth B. Elliott, III

Cynthia L. Glabus

Dana Nell Stigdon

Michael J. Oehler

Cornelius J. Haley

The team members were provided by the following organizations:

National Computer Security Center
Fort George G. Meade, Maryland

The Aerospace Corporation
Los Angeles, California

Further Acknowledgements

Technical support was also provided by Kevin Curley of the International Business Machines Corporation.

Table of Contents

This page intentionally left blank.

EXECUTIVE SUMMARY

The security protection provided by the International Business Machines Corporation Multiple Virtual Storage/Extended Architecture (MVS/XA) operating system with the Resource Access Control Facility (RACF) product (see page B-1, "Appendix B, Evaluated Software Components"), configured according to the most secure manner described in the Trusted Facility Manual, running on System/370 Extended Architecture (XA) machines (see page A-1, "Appendix A, Evaluated Hardware Components") has been examined by the National Computer Security Center (NCSC). The security features of MVS/XA with RACF were evaluated against the requirements specified by the DoD Trusted Computer System Evaluation Criteria (the Criteria) dated December 1985.

The NCSC evaluation team has determined that the highest class at which MVS/XA with RACF satisfies all the specified requirements of the Criteria is Class C2 and therefore, using the specified hardware, MVS/SP Version 2 Release 2 with RACF Version 1 Release 8 configured in the most secure manner described in the Trusted Facility Manual, has been assigned a class C2 rating.

A system that has been rated as being a C division system provides for discretionary protection and, through the inclusion of audit capabilities, accountability of subjects and the actions they initiate. Class C2 systems enforce a finely grained discretionary access control, making users individually accountable for their actions through login procedures, auditing of security-relevant events, and resource isolation.

MVS/XA is IBM's flagship operating system for IBM 3090 Series E and IBM 4381 mainframe computers supporting the System 370 Extended Architecture. MVS/XA adheres to this architecture taking advantage of the various protection mechanisms offered. MVS/XA systems may execute on machines with up to six processors, which themselves may be combined with other MVS/XA systems to form MVS/XA complexes capable of supporting hundreds or thousands of users. MVS/XA with RACF can be used in a wide variety of environments.

The MVS/XA operating system is a general purpose time-sharing and batch system with several security features. The hardware provides address space separation and privilege isolation. The RACF product enhances certain security features providing a controlled access to system resources using access control lists, a default protection of these resources, an extensive auditing facility, and a role separation among privileged users.

This page intentionally left blank.

INTRODUCTION

In August 1987 the National Computer Security Center (NCSC) began a formal product evaluation of Multiple Virtual Storage/System Product (MVS/SP) Job Entry Subsystem 2 (JES2) Version 2 Release 2 with Resource Access Control Facility (RACF) Version 1 Release 8. Other products included in this evaluation were: Data Facility Product (DFP) Version 2 Release 3, Advanced Communications Function/Virtual Telecommunications Access Method (ACF/VTAM) Version 3 Release 1.1, and Time Sharing Option/Extensions (TSO/E) Version 1 Release 4. All of these items are products of International Business Machines Corporation (IBM). The objective of this evaluation was to rate the MVS/XA system against the Criteria, and to place it on the Evaluated Products List (EPL) with a final rating. This report documents the results of that evaluation. This evaluation applies to the system as available from IBM in June 1988.

Material for this report was gathered by the NCSC MVS/XA evaluation team through documentation, interaction with system developers, and experience using MVS/XA systems.

Evaluation Process Overview

The Department of Defense Computer Security Center was established in January 1981 to encourage the widespread availability of trusted computer systems for use by facilities processing classified or other sensitive information. In August 1985 the name of the organization was changed to the National Computer Security Center. In order to assist in assessing the degree of trust one could place in a given computer system, the DoD Trusted Computer System Evaluation Criteria was written. The Criteria establishes specific requirements that a computer system must meet in order to achieve a predefined level of trustworthiness. The Criteria levels are arranged hierarchically into four major divisions of protection, each with certain security-relevant characteristics. These divisions are in turn subdivided into classes. To determine the division and class at which all requirements are met by a system, the system must be evaluated against the Criteria by an NCSC evaluation team.

The NCSC performs evaluations of computer products in varying stages of development from initial design to those that are commercially available. Product evaluations consist of a developmental phase and a formal phase. All evaluations begin with the developmental phase. The primary thrust of the developmental phase is an in-depth examination of a manufacturer's design for either a new trusted product or for security enhancements to an existing product. Since the developmental phase is based on design documentation and information supplied by the industry source, it involves no "hands on" use of the system. The developmental phase results in the production of an Initial Product Assessment Report (IPAR). The IPAR documents the evaluation team's understanding of the sys'em based on the information presented by the vendor. Because the IPAR contains proprietary intormation, distribution is restricted to the vendor and the NCSC.

Products entering the formal phase must be complete security systems. In addition, the release being evaluated must not undergo any additional development. The formal phase is an analysis of the hardware and software components of a system, all system documentation, and a mapping of the security features and assurances to the Criteria. The analysis performed during the formal phase requires "hands on" testing (i.e., functional testing and, if applicable, penetration testing). The formal phase results in the production of a final report and an Evaluated Products List entry. The final report is a summary of the evaluation and includes the EPL rating which indicates the final class at which the product successfully met all Criteria requirements in terms of both features and assurances. The final report and EPL entry are made public.

MVS/XA with RACF Background and History

MVS has evolved from over twenty years of operating system design. Its development began in 1964 with the announcement of the System/360 computers and was to be IBM's solution to incompatible system software for different computer systems. The six System/360 computers used a standard architecture and instruction set to provide compatibility for a wide range of computer systems.

OS/360 was a single operating system designed to serve all System/360 computers. There were three versions of OS/360: Primary Control Program (PCP), Multiprogramming with a Fixed number of Tasks (MFT), and Multiprogramming with a Variable number of Tasks (MVT). The three versions were similar in that they shared the same Job Control Language (JCL), but differed in job scheduling and resource management. PCP was an operating system designed to control a single program at a time, whereas MFT and MVT offered multiprogramming.

MFT used a single real address space which consisted of a nucleus residing in low storage and up to fifty-two fixed sized partitions. There were two types of partitions, system and problem state. User jobs executed in a problem state partition containing an active initiator and were assigned one of the 4 bit protection keys (key 0 was used in system partitions). Since there were only fifteen available keys, MFT could control a maximum of fifteen multiprogrammed jobs.

In many ways, MVT was similar to MFT. MVT controlled job processing with JCL, used protection keys to assign jobs to a class, and contained many of the same system operations (e.g. a nucleus, Master Scheduler, and Link Pack Area) as MFT. However, MVT did not fix the sizes of its partitions but rather varied them dynamically based on allocation requests it received from job initiators.

In 1972 IBM announced virtual storage (VS) for many of their System/370 computers. The VS function could be incorporated into existing System/370 computers by adding Dynamic Address Translation (DAT) and Translation Lookaside Buffer (TLB) hardware, adding microcode (found on smaller systems), or bought outright on newly introduced machines. Although the hardware and its speed varied for each machine, the logical function of DAT did not change.

Based upon OS/360, IBM created three operating systems for the System/370 VS computers: Operating System/Virtual Storage 1 (OS/VS1), Operating System/Virtual Storage 2 Release 1 (OS/VS Rel 1), and OS/VS2 Rel 2, known as MVS. IBM also offered DOS/VS and Virtual Machine/370 for the System/370 computers, but these operating systems were developed from DOS and CP/67, respectively. OS/VS1 was the VS successor of MFT and offered fixed segments in a single-virtual 16M address space.

The two releases of OS/VS2 were significantly different. OS/VS2 Rel 1 was the VS replacement for MVT. OS/VS2 Rel 1 was similar to MVT in that it dynamically allocated space to jobs, but different in that it had a 16M virtual address space and allocated space in 64K segments. This allocation scheme was used so that OS/VS2 Rel 1 could incorporate paging, but resulted in fragmentation of the virtual address space. A more advanced approach was used by OS/VS2 Rel 2 *which had multiple virtual address spaces.* OS/VS2 Rel 2, known as MVS, became available in 1974 and has undergone two product revisions since then. The first revision, MVS/SP 1 was announced in 1978 and was followed by MVS/XA, the extended architecture version, in 1982.

In addition, IBM has developed programs which, when used with the operating system, serve and assist both the system and its users. Each one of these programs has a dedicated task. A brief background is given here for three of these products: Job Entry Subsystem (JES), Time Sharing Option/Extensions (TSO/E), and Resource Access Control Facility (RACF). Further descriptions of these programs can be found in the corresponding sections of the report.

JES is a system component responsible for the job scheduling and removal of job output. JES has developed from the Houston Automatic Spooling Priority (HASP) which was developed by a user group. HASP improved the standard OS/360 (i.e. MFT and MVT) I/O spooling and provided a dispatching option to improve the throughput of I/O bound jobs. Although the capabilities of JES have been enhanced over that of HASP, similarities are present.

The Time Sharing Option (TSO), offered in 1971, provided the OS/360 user with the system's first full featured timesharing capabilities. There were other timesharing systems for IBM systems, for example Compatible Time-Sharing System (CTSS) developed by MIT, Time Sharing System by IBM, and IBM's A Programming Language. However, many of these were dedicated to certain applications, whereas TSO provided interactive access to language development facilities, command processing, utilities, and (background) batch job processing. Many of these functions were not offered in the standard batch environment or any one timesharing program. TSO has been enhanced to include several additional features (referred to as TSO/Extensions).

RACF was announced in 1976 and provided control for user identification and authorization, access control, and auditing. RACF was developed from several prototype programs in order to enhance system security. The original prototype program provided additional catalog information for data sets and extended cataloging capabilities.

IBM has been developing MVS for the past twenty years and the most current version (MVS/SP Version 2 Release 2) has incorporated many of the capabilities, applications, and performance factors of former systems. Because of this development, MVS/XA offers a level of compatibility among MVS systems and backward compatibility with former systems.

Document Organization

This report consists of eight sections and three appendices. Section 1 is an introduction. Sections 2 and 3 are an overview of the hardware and software architectures respectively. Section 4 is a description of the protected resources in MVS/XA with RACF. Section 5 is a description of the protection mechanisms available in MVS/XA with RACF. Section 6 is a description of the assurances provided by IBM for MVS/XA with RACF. Section 7 is a mapping of MVS/XA with RACF protection mechanisms to the C2 requirements in the *DoD Trusted Computer System Evaluation Criteria.* Section 8 contains the evaluator comments. The three appendices identify the hardware and software components of an evaluated system, and the acronyms used throughout the report.

HARDWARE ARCHITECTURE

INTRODUCTION

The MVS/XA with RACF system runs on machines with the System 370-XA architecture. Certain models have a vector processing capability, and these models are included in the evaluated configuration. The mainframes included in this evaluation are the 4381 and the 3090 series of machines (for specific model numbers, see page A-1, "Evaluated Hardware Components"). These machines all possess the Extended Architecture (XA) mode of operation which is specified in the *IBM System 370 Extended Architecture Principles of Operation* manual and described briefly in the following section.

The system consists of main storage, one or more central processors (CPs), operator facilities, a channel subsystem, control units, and Input/Output (I/O) devices. Main storage, which is directly addressable by the CPs, provides for high-speed processing of data by the CPs and the channel subsystem. The central processor contains the sequencing and processing facilities for instruction execution, interruption action, timing functions, and other machine-related functions.

Each CP provides registers which are available to programs but do not have addressable representation in main storage. They include the current Program Status Word (PSW), sixteen 32-bit general registers, four 64-bit floating-point registers, sixteen 32-bit control registers, the prefix register, and the registers for the clock comparator and the CP timer.

The channel subsystem directs the flow of information between I/O devices and main storage. I/O devices are attached through control units to the channel subsystem via channel paths. Control units may be attached to the channel subsystem via more than one channel path, and an I/O device may be attached to more than one control unit.

In addition to the protection mechanisms listed below, the System 370-XA architecture provides two states of operation, supervisor and problem (user). This is controlled in the Program-Status Word (PSW). When in supervisor state, the currently executing process may use privileged instructions. These abilities allow the current process the ability to circumvent all security mechanisms. Therefore, all programs with the ability to run in supervisor state are part of the Trusted Computing Base (TCB). When in problem state, processes are restricted by the memory protection mechanisms and are not allowed to execute privileged instructions. These restrictions cause problem-state processes to abide by the security policies enforced by the TCB. However, problem state programs can run with keys 0-7 (see page 12, "Key-Controlled Protection") if APF authorized (see page 54, "Authorized Programs").

Programs change from problem state to supervisor state only through Supervisor or Program Call instructions, or an interrupt. Both of these cause a new PSW to be loaded from a protected storage location (see page 12, "Key-Controlled Protection").

There are two families of machines that are included in the evaluated configuration: the 3090 family and the 4381 family. The following sections describe, in detail, the specifics of the architectures of each family, and the differences in the models listed in Appendix A (see page A-1, "Evaluated Hardware Components").

MODEL 3090 PROCESSORS

The models of the 3090 family that are in the evaluated configuration fall into two categories: those with one CP and those with more than one CP. Model numbers prefixed with a "1" (e.g., 120, 180) fall into the former category, while the 200, 300, 400, 500, and 600 refer to the 2, 3, 4, 5, and 6 processor models, respectively.

A 3090 uniprocessor has central storage, expanded storage, subchannels, a system control element, and a central processor. Subchannels and expanded storage are described in later sections in the overview.

The system control element (SCE) has the capability to control up to three processors. While the uniprocessor models only have one SCE, the 400, 500, and 600 models have two SCEs, each of which controls 2 or 3 CPs. Each SCE communicates directly with its own central and expanded storage, subchannels, and CPs, as well as with the other SCE (if another exists). The 200 and 300 models only have one SCE with two and three CPs. Central storage is composed of up to four processor memory arrays (PMA), with a maximum of two being controlled by one SCE. Expanded storage is composed of up to four expanded storage arrays (ESA), again with a maximum of two being controlled by one SCE. Expanded storage is a solid-state memory array that can only be addressed in 4K sections, called pages. It can transfer these pages between itself and the central storage in a synchronous manner with respect to the CP. It has no designed in hardware functions, but MVS/XA uses it as a solid-state paging device. All accesses to resources on the other side (i.e., a CP accesses a resource that is controlled by the other SCE) will go through both SCEs, causing a slight delay. The expanded storage is addressed through the use of special op codes. These op codes can only be issued by authorized programs, and are in fact only issued by the Real Storage Manager and the Auxiliary Storage Manager in the MVS system.

There is also a Processor Control Element (PCE), which is the hardware controller for the model 3090 machines. The PCE has two distinct sides, which can either act as a primary-backup pair, or a primary-primary pair. Each of these sides has a directly attached 3370 DASD unit, which is used to contain information about the system. When in primary-backup mode, the data is shadowed from the primary to the backup. When in primary-primary mode, no data is shadowed.

The PCE is a separate physical unit communicating with the 3090 through lines from PCE-based Logic Support Adaptors (LSA) to 3090 component-based Logic Support Stations (LSS). Each LSA controls an LSS. An LSS is contained in every module of the 3090 hardware (e.g., SCE, main memory, etc.).

The 400 and 600 models have the ability to be sectioned by the hardware into two distinct sections, which have no electronic communication between them. Hence, a 400 could be partitioned so that it appeared to those running on it as two 200s, and likewise a 600 would appear as two 300s. This is called physically partitioned (PP) mode. When this is not in effect, the configuration is referred to as running in single image (SI) mode.

A machine can be put into PP mode (or taken from PP mode to SI mode) only from the system or service consoles, which are attached directly to the PCE (see page 40, "Consoles"). These are distinct from the operator's console, which is the console from which MVS/XA is controlled (see page 40, "Consoles").

When a machine is put into PP mode, the side of the machine to be partitioned must be taken offline. If this is not done, the partitioning operation will not be allowed. Next, the backup side of the PCE will be assigned as primary to the new side. After a power-on reset, the new side is available for use.

The separation is achieved by way of the LSSs. There are 4 bi-directional buses between the two sides: three between the two SCEs, and one between the two expanded storage controllers (each SCE has its own expanded storage controller, which is responsible for both banks of ESA associated with that SCE). In addition to the buses, there are associated control and status lines. When the PP mode is activated, a bit in the LSS is zeroed. This bit is ANDed with all of the lines going between the two sides, including the data ready lines. This makes it appear to one side that the other side does not exist.

A bit in the LSA determines which side of the PCE it is assigned to. When powered up, all LSAs are assigned to one side, and the other side becomes the backup. This is determined by a switch on the front panel. When going to PP mode, the LSAs responsible for the LSSs on the offgoing side are reassigned to the (new) primary side. For instance, if A was the primary and B was the backup, when going to PP mode, half of the LSAs would be assigned to the B side from the A side.

There is no way for the 3090 to tell the PCE side controlling its LSSs to switch them on or off, and one side of the PCE (when in PP mode) can not tell the other side to switch its LSAs; this must be done by the side that owns the LSAs.

When the 3090 is physically partitioned into two separate machines, one side may run an evaluated system, while the other may run any kind of system. Both sides may run separate evaluated (MVS/XA with RACF) systems, but if DASD devices are shared, then both systems must adhere to the complex restrictions described on the next page.

The design of the 3090 allows growth in a number of areas. The system is designed so that there can be a maximum of 256 channels, 64K devices, 16 processors, and 2 GB of central memory. At present, a maximally configured system contains 128 channels, 4080 devices, 6 processors, 256 MB of central storage, and 2GB of expanded storage.

MODEL 4381 PROCESSORS

The 4381 series of processors are an evolution of the 4341 processors, and are IBM's mid-range family of mainframes, falling between the 9370 family (non-XA) on the low end and the 3090 family on the high end. There are both single and dual processor versions of this machine; the model 14 and model 24 are dual processors, while other models are single processor.

Each processor communicates with a common memory subsystem, while each processor communicates with its own channel subsystem. The two CPs can communicate through a support subsystem. The maximum memory configuration on a 4381 machine is 64 MB, and each CP can communicate with a maximum of 24 channels. Models prefixed with a "2" are IBM's next generation of the 4381 series of machines, primarily incorporating speed and size (memory and channels) enhancements.

JES COMPLEXES

In addition to the single machine configurations described above, configurations comprising more than one machine are also supported. These configurations are known as JES complexes, or just as "complexes". In order for complexes to be considered as a valid configuration, some restrictions must be in force. A description of the complex itself and restrictions needed for a C2 system are provided below.

A complex can be composed of any number (up to 7) machines running a copy of JES (see page 55, "JOB ENTRY SUBSYSTEM 2"). Complexes included in the evaluated configuration will only be composed of those mainframes listed on page A-1, "Evaluated Hardware Components". All machines will be running instantiations of one copy of the TCB software, which is listed on page B-1, "Evaluated Software Components".

Additional hardware for complexes in the evaluated configuration include Channel-to-Channel adaptors (CTCs) and shared DASD. CTCs are devices that allow the JES running on one machine in a complex to communicate with the JES running on another machine in the complex. The CTC appears as an I/O device to each of the JESs, and is addressed as such. The shared DASD is needed for those data sets which must be accessed by more than one machine in the complex. The controller for a shared DASD mediates this sharing.

Machine IDs in the complex are defined to JES. A valid complex must have every machine in the complex defined to JES and GRS; and all machines must use the same RACF database. Subsetting is not allowed. All shared data set use must be synchronized under control of the DASD on which these data sets reside. Synchronization mechanisms include RESERVE-DEQ (see page 47, "Serializing Resources") and GRS (see page 49, "Global Resource Serialization").

Interactive users cannot interactively choose which machine in the complex they will log into; terminals are defined via VTAM (see page 68, "ACF/VIRTUAL TELECOMMUNICATIONS ACCESS METHOD") to one machine at Initial Program Load, and cannot be changed by an untrusted user. Batch jobs will be selected off the queue of ready batch jobs when an initiator is ready (see page 38, "MVS/XA Initiation"), regardless of the machine the initiator is on. Each of the phases of job execution (see page 57, "Execution") can run on a different machine in the complex, but a given phase runs to completion on one machine.

In addition to the RACF database, all JES spool data sets must be shared across the complex, as job phase scheduling is done on a complex-wide basis by the JESs. The checkpoint data set for job restart in case of failure must also be complex-unique. Data sets that cannot be shared among machines in the complex include each machine's paging data set, system log, and dump data set. Each machine in the complex also has its own copy of SMF (see page 107, "System Management Facilities") and associated audit data set, and audit data is collected for each machine with the machine ID, time, and user included in the audit record. Facilities exist for merging the audit data sets before the audit reduction tool is used in order to provide a comprehensive audit trail (see page 110, "Complex Auditing").

ADDRESS SPACE SELECTION

The MVS/XA with RACF operating system, in combination with the System 370-XA architecture, supports virtual addressing for all address spaces existing on the system. The layout of the address space is discussed in the MVS Overview section of TCB Architecture (see page 28, "MVS/XA STRUCTURE"). This section describes hardware management of address space switching. The mechanisms described here are used primarily for context switching and establishing new address spaces.

Each address space in the system is assigned an address space number (ASN); there can be up to 64K different address spaces in the system. A given processor can address two different address spaces without going through an address space translation process (as opposed to virtual address translation, described later). These two address spaces are called the primary and the secondary, and selection is controlled by bit 16 in the PSW. In MVS/XA, this mechanism is used to provide cross memory services (see page 30, "Cross Memory Services"). An address space is defined by the segment table (see page 11, "DYNAMIC ADDRESS TRANSLATION") for that address space; the segment table addresses for the primary and secondary address spaces are located in control registers 1 and 7, respectively. Since the XA architecture supports 31-bit virtual addressing (previously, System/370 could only support 24-bit addressing), the size of each of the address spaces is about 2 GB.

In order to swap a primary or secondary address space, the system uses a number of tables to perform a translation from the 16-bit address space number to the address of the segment table representing that address space. Two tables, the ASN first table and ASN second table, in conjunction with the ASN authority table, are used to perform this translation. Control register 14 contains the address of the ASN first table in bits 13 - 31. The first 10 bits of the ASN are an index into this table; the selected entry gives the address of the ASN second table entry. The final 6 bits of the ASN are an index into the ASN second table entry.

The ASN second table entries are made up of a number of fields: the address for the authority table, the length of the authority table, and index into the authority table, an address for the segment table for this address space, and linkage table information which goes in control register 5.

The authority table is made up of groups of four 2-bit entries, with the first bit designated as the primary authority (authority to use the address space as a primary address space) and the second bit designated as the secondary authority (authority to use the address space as a secondary address space). Control register 4 contains a 14-bit index into the authority table. Bits 14 and 15 indicate which 2-bit section of the designated authority table entry to use in determining the authority. If the selected bit is 1, then the ASN is authorized; otherwise, it is not authorized. As in all cases where an address space translation encounters a disallowed condition, an interrupt is taken, and the translation and subsequent switch is not completed.

DYNAMIC ADDRESS TRANSLATION

In order for a virtual address to be translated to an address in real memory, the System 370-XA architecture provides the DAT mechanism. A virtual address space is divided into segments and pages; there are 2048 segments (1 Megabyte (MB) of data), each with 256 4K pages.

To translate a virtual address to a real address, the segment table address is obtained from control register 1 or 7. Bits 1-11 of the virtual address form the index into the segment table for the desired page table address. Bits 12-19 are the offset for the page table entry in the page table, which designates a page frame. The final bits (20-31) of the virtual address give the offset of the desired address in the page frame. Each of the page table entries has an invalid bit, used to detect page faults, and a protection bit, which is described below.

Each CP has associated with it a translation-lookaside buffer (TLB). The purpose of this buffer is to speed up DAT. The size and implementation of the TLB is model dependent. When a virtual address is translated (using DAT) to a real address, all the information used in this translation, including protection information such as the page-protection bit from the page table entry, is conceptually placed in this buffer. When the page is again referenced, the TLB can be used to locate the page in a much faster manner than with the normal DAT. Rules for placing information in the TLB are complex, and are treated fully in the *IBM System 370 Extended Architecture Principles of Operation*

PREFIXING

In multi-CP systems, the different CPs sometimes need to access common low storage (address locations 0-511) values (e.g., interrupt vectors) at the same time. To make this operation more efficient, the System 370-XA architecture includes the concept of a prefixed save area, which is the first 4K bytes of real storage. In the following discussion, absolute addresses are those address that are assigned to each physical memory location. Real addresses are those addresses which are used to access memory (i.e., result of virtual address translation), and correspond exactly to absolute addresses except as outlined below.

Each processor has a prefix register, which is used to determine the absolute address of real addresses 0-4095. The block of real addresses specified by the value in the prefix register correspond to absolute addresses 0-4095. This allows processors to access each others' prefixed save area. Except for these two blocks of real addresses, all other real addresses correspond to the absolute addresses

HARDWARE PROTECTION MECHANISMS

The System 370-XA architecture and the MVS/XA with RACF operating system provide three separate protection mechanisms for data in main storage: key-controlled protection, page protection, and low-storage protection. In order for a process to access any location in storage, the checks from all three of these mechanisms must allow the access. These checks are performed after address translation (i.e., they are based on physical memory locations).

Key-Controlled Protection

Key-controlled protection provides protection against unauthorized storing, or unauthorized storing and fetching for locations in main memory. Memory locations can not be fetch protected only (i.e., can not be protected from reading while allowing writing). Each 4K page of real storage has associated with it a 7-bit storage protect key composed of four access control bits (the protect key), a fetch-protect bit, a reference bit, and a change bit. These keys, while associated with every page of real storage, are not a part of addressable storage, and are set with privileged instructions (see page 14, "PRIVILEGED and SEMI-PRIVILEGED INSTRUCTIONS"). Whenever a program attempts to access a storage location, a comparison is made between the protect key on the page in storage and an access key that the program possesses. This access key is located in the PSW for programs running on a CP, and in the operation request block (see page 33, "I/O Operations") for channel programs. If the keys match, then the requested access is allowed. If the keys do not match, a protection exception interrupt occurs and the instruction is terminated. This mechanism is applied to both supervisor state and problem state programs.

There are several instances when key-controlled protection is not active. They are:

- Processing an interrupt.
- Fetching page table entries for DAT or ASN translation.
- Tracing program execution.
- A store-status function (used to store register and timer values to low storage).
- Initial program load.
- Operator functions.

The following are the key assignments for the system. Those marked with an asterisk (*) are used in products found in the evaluated configuration.

Key 0* Key 0 is used for the MVS/XA system control program.

Key 1* Key 1 is used for the Job Scheduler and Job Entry Subsystem (JES2).

Key 2 Key 2 is used for VSPC (virtual storage personal computing).

Key 3,4 Reserved.

Key 5* Key 5 is used for data management (DFP).

Key 6* Key 6 is used for TCAM/VTAM (terminal access to system).

Key 7 Key 7 is used for IMS.

Key 8* Key 8 is the normal user key. All users running in virtual address spaces run with key 8.

Keys 9-15* Keys 9-15 are used when a user process must run in V=R mode (see page 28, "Address Spaces").

Page Protection

Page protection controls the storing of data in virtual memory. This is done by means of a bit in each page table entry. If the bit is set, the page is read-only. If the bit is not set, read and write access are both allowed. Attempted writes to page protected portions of memory cause a protection exception interrupt to occur, and the write does not take place. The page tables themselves are protected by the key-controlled protection mechanism.

Low-Address Protection

The low-address protection mechanism prevents code from modifying certain critical locations in low-storage which contain information for the processing of exceptions and interrupts. If the low-address-protection-control bit is set, low address protection is in effect and any attempted write access to the protected memory location will cause a protection exception interrupt to occur and the instruction to be terminated. Low-address protection does not apply to the CP or the channels when they are processing an interrupt (e.g., when fetching a new PSW).

PRIVILEGED and SEMI-PRIVILEGED INSTRUCTIONS

There are two types of security relevant machine language instructions for the System 370-XA architecture: privileged and semi-privileged. Privileged instructions are those instructions that can only be issued while the CP is in supervisor state. Semi-privileged instructions can be issued from either state, but have other restrictions applying to their use. The I/O instructions are also privileged. The following table gives the privileged instruction mnemonics and brief description of each.

DIAGNOSE	This instruction has no mnemonic. Its arguments are a code that hasdifferent meaning to different models in the product line.
ISKE	Insert storage key extended - put the storage key for the designated block of real storage into a general register.
IPTE	Invalidate page table entry - invalidate the designated page table entry, and clear associated entries in the TLBs of all CPs.
LASP	Load address space parameters - This instruction loads various valuesused in address space manipulations. The four operations performed by this instruction include primary ASN translation, secondary ASN translation,secondary ASN authorization, and control-register loading (can affectregisters 1, 3, 4, 5, and 7).
LCTL	Load control - This instruction controls loading of any contiguous block ofcontrol registers.
LPSW	Load PSW - Replaces the current PSW with the specified PSW.
LRA	Load real address - Loads the real address corresponding to the designatedvirtual address into the specified general register.
PTLB	Purge TLB - This instruction purges the TLB of all entries for the issuing CP.
RRBE	Reset reference bit extended - This instruction sets the reference bit in thestorage key for the designated page to zero.

SCK Set clock - This instruction sets the Time-of-Day (TOD) clock to thespecified value, and stops the clock. Starting the clock is dependent upon a bit in control register 0 being set.

SCKC Set clock comparator - Set the clock comparator to the designated value.

SPT Set CP timer - Set the value of the CP timer to the designated value.

SPX Set prefix - This instruction sets the prefix register of the issuing CP to the designated value, and has the side effect of clearing that CP's TLB.

SSKE Set storage key extended - This instruction sets the value of the storage key of the designated page to the value specified.

SSM Set system mask - The system mask refers to bits 0-7of the PSW;this instruction sets those bits to the designated value.

SIGP Signal processor - Send an 8-bit order code, and possibly a 32-bit parameter to the designated CP.

STCKC Store clock comparator - This instruction reads the clock comparator,and stores the value at the designated address.

STCTL Store control - This instruction reads the values of the designatedcontiguous set of control registers, and stores these values at the designated address.

STAP Store CP address - Every CP in a multi-processor configuration isidentified by an address. This instruction stores the address for theissuing CP at the address specified.

STIDP Store CP ID - The ID of an individual CP consists of a version code, ID number, and a model number. This information is stored at the designated address.

STPT Store CP timer - This instruction reads the value of the CP timer into thedesignated address.

STPX Store prefix - This instruction reads the value of the issuing CP's prefix register into the address specified.

STNSM Store then AND system mask - This instruction saves bits 0-7 of the PSW in a specified address, logically ANDs bits 0-7 of the PSW with the second operand, and stores the result back to bits 0-7 in the PSW.

STOSM Store then OR system mask - Same as above, except a logical ORis performed.

TB Test block - Tests the usability of the locations and the storage key associated with the designated 4K block (page) of storage, based on the susceptibility of the block to the occurrence of invalid checking-block code.

TPROT Test protection - This instruction tests the designated address for any protection exception that would occur with the access key given in the second operand.

TRACE Trace - This instruction is used to form trace entries when tracing is turned on.

The following table gives the privileged instruction mnemonics and brief description of each.

CSCH Clear Subchannel - This instruction clears the designated subchannel, and signals the channel subsystem (asynchronously) to perform a clearfunction at the associated devices.

HSCH Halt Subchannel - This instruction signals the channel subsystem to terminate the current start function at the designated subchannel and associated devices.

MSCH Modify Subchannel - This instruction causes the information in the subchannel information block into the appropriate program-modifiable fields of the subchannel itself. These fields influence clear, halt, resume, and start functions of the subchannel, as well as certain I/O support functions.

RCHP Reset Channel Path - This instruction signals the channel-path-reset facility to perform a reset on the designated channel path.

RSCH Resume Subchannel - This instructions signals the channel subsystem to resume operations at the designated subchannel.

SAL Set Address Limit - This instruction passes an address limit to theaddress-limit-checking facility for use in checking addresses for an out-of-bounds condition.

SCHM Set Channel Monitor - This instruction sets the monitoring modes of the channel subsystem either active or inactive, depending on the contents of general register 1.

SSCH Start Subchannel - This instruction places the contents of the Operations Request Block (ORB; see page 33, "I/O Operations") in the subchannel, and signals the subchannel to perform the start function.

STCPS Store Channel Path Status - This instruction places up to 256 bits of information which reflects the active channel paths for that subchannel into the designated location.

STCRW Store Channel Report Word - This instruction places a channel report word, which contains information about conditions affecting the channel subsystem (e.g., malfunction), into the designated location.

STSCH Store Subchannel - This instruction places control and status information from the designated subchannel into a subchannel information block.

TPI Test Pending Interruption - This instruction stores the code for apending interruption at the subchannel in the designated location, and clears the interrupt request.

TSCH Test Subchannel - This instruction stores control and status information from the designated subchannel into an interrupt request block.

The following table lists the mnemonic, function and privilege(s) needed for the semi-privileged instructions.

EPAR Extract primary ASN - The primary ASN is placed in the designated general purpose register. Bit 4 of control register 0 (the extraction authority control) must be 1 in the problem state.

ESAR Extract secondary ASN - The secondary ASN is placed in the designated general purpose register. Bit 4 of control register 0 (the extraction authority control) must be 1 in the problem state.

IAC Insert address space control - Bit 16 of the PSW designates which of the address spaces (primary or secondary) will be used for DAT. This instruction places the bit in the designated general purpose register. Bit 4 of control register 0 (the extraction authority control) must be 1in the problem state.

IPK Insert PSW key - This instruction inserts the PSW access key (bits 8-11) into the designated general purpose register. Bit 4 of control register 0 (the extraction authority control) must be 1 in the problem state.

IVSK Insert virtual storage key - This instruction puts the storage key for the location designated by the virtual address into the specified general purpose register. Bit 4 of control register 0 (the extraction authority control) must be 1 in the problem state.

MVCP Move to primary,

MVCS Move to secondary - MVCP moves the data at the specified address address space. MVCS moves the data from the primary address space to the secondary address space. In the problem state, the PSW key mask bit (in control register 3) must be one. Also, bit 5 of control register 0 (secondary space control) must be set.

MVCK Move with key - This instruction performs a move from one memory address to another. The memory address of the location the data is being moved from is checked with a key that is specified in the instruction; the memory location that the data is being moved to is checked with the PSW access key (in control register 3). In the problem state, these bits must match.

PC Program call - This instruction is used to transfer execution to another location in either the current address space or a different address space. Transfer is controlled by entry table. The entry table contains a field which, when the CP is in problem state, is checked against the current PSW-key mask to assure authorization for the program making the call. Various control bits must also be set before this instruction succeeds.

PT Program transfer - This instruction is similar in effect to PC above, but takes values for the various registers needed on a switch from the two operand registers. If a switch from problem state to supervisor state occurs, an exception results. For space switching forms of this request, the authorization codes (described above) must match for successful execution.

SAC Set address space control - This instruction sets the address space (primary or secondary) to be used. Bit 5 of control register 0 must be set, and DAT must be on for this instruction to successfully execute.

SPKA Set PSW key from address - Bits 8-11 of the PSW are replaced by bits 24-27 of the designated address. The PSW key value must match the PSW key mask in control register 3 in order for the instruction to be completed in problem state.

SSAR Set secondary ASR - This instruction sets the secondary address space to the address space designated by the ASR given in the operand. Bit 12 of control register 14 must be set, and DAT mustbe on, for this instruction to successfully execute in problem state.

INPUT/OUTPUT

I/O in System 370-XA architecture machines is performed at the hardware level by the channel subsystem, the control units, and the I/O devices. A description of a full I/O operation, including the use of Channel Control Words (CCWs), is presented later (see page 33, "I/O Operations"). In addition to this "traditional" I/O method, IBM also supports Virtual I/O, which is described in the MVS discussion (see page 35, "Virtual Input/Output").

The Channel Subsystem

The channel subsystem, pictured in figure 1, directs communication between the I/O devices and main storage. In this figure, the CUs represent control units, and the Os represent devices. The channel subsystem mediates communications between the I/O devices and storage, freeing the CP so that I/O and data processing can take place concurrently. The subsystem is composed of different "subchannels." There can be up to 64K subchannels, and each subchannel is uniquely associated with a single device. A device, however, may be associated with a number of different subchannels. A subchannel is the addressable unit designated by a program in initiating I/O operations.

Subchannels are the logical abstraction of the hardware Channel Control Elements (CCE) and the actual channels. There can be up to two CCEs per machine, depending on the model. A CCE can control up to 16 Channel Elements, each of which has four physical channels. A maximally configured 3090 model 600 can contain 128 channels. Technically, a channel subsystem is one CCE with its associated channels, but in machines with two CCEs that are not partitioned, the two channel subsystems coordinate activity and appear to the control program as one dynamic channel subsystem. The CCE is the physical unit that implements the subchannel, holding all information necessary to define that subchannel.

There are two types of channels: byte multiplexer and block multiplexer. Byte multiplexer channels can be shared by many devices, or dedicated to one device. For 4381 machines, there can be a maximum of two byte multiplexer channels. On the 3090 series, the 400 and 600 models can have up to eight channels configured as byte multiplexers, while other models can only have a maximum of four.

Block multiplexer channels are slightly different for 4381 machines and 3090 machines. For 4381s, there are two modes for block multiplexers: block multiplexer mode, and selector mode. Block multiplexer mode allows more than one high-speed I/O device to use the channel, while selector mode restricts the channel to one I/O device until the I/O operation is complete. 3090 block multiplexer channels also have two modes: interlocked and data streaming. Both modes are similar to the 4381 selector mode, in that the device and channel are connected throughout the life of the

I/O operation. Interlocked mode uses interlocking data transfer signals between the channel and the control unit, while the data streaming mode does not. Consequently the data streaming mode is much faster than the interlock mode (4.5 MB per second, as opposed to 1.5 MB per second).

The channel subsystem contains the necessary logic and storage to direct the I/O between the device and main storage. To mediate the transfer, the subsystem uses control units, which lie between the subsystem and the device, and channel paths to these control units. An I/O device can be connected to up to eight control units; the control units in turn are connected via channel paths to the subsystem. There can be more than one channel path to each of the control units, depending on the model of the control unit (See page A-1, "Evaluated Hardware Components"). The maximum number of different channel paths that can be addressed by the control unit is 256. The control unit is responsible for converting the generic request issued by the subchannel to a request that can be recognized by the device. When an I/O request is given, an available channel path is chosen. This means that two successive reads from the same data set need not take the same channel path to the device, although the subchannel number would be identical for both reads.

Control Units and Devices

Many different devices are supported in the evaluated configuration. The specific model numbers are listed in Appendix A, but general architectural details of the "intelligent" control units and devices are discussed in this section.

DASD Controllers

There are two main models of DASD controllers in the evaluated configuration: the 3880 and the 3990. The 3880 is the older of the two models. The model 3880-3 can have two paths to a DASD, meaning it can support two "strings" (strings of DASD are described below). The 3880-23 is identical to the 3880-3, except that it contains a cache to buffer data coming off of the drive. The 3880-21 is only used for paging, and must be connected to the 3350 model DASD. Since the 3880 has two paths, two of these controllers can be cross connected with two strings of DASD, allowing two paths to the same string from different control units. This redundancy increases the reliability of the system.

The 3990-1 is a two path version of this model line, while the -2 and -3 support four paths to DASD. In addition, the 3990-3 can have a cache of up to 256 MB, which decreases the average access time from 22-32 ms to 3-5 ms. All of the 3990 controllers can be cross-configured as detailed in the 3880 discussion.

DASD Disks

The 3380 DASD devices in the evaluated system are divided into 2 families: the D/E family, capable of supporting 2 paths per string; and the J/K family, capable of supporting 4 paths per string (with the 3990 controller). Each DASD has two disks per cabinet. D and J type 3380s can store 1.25GB per disk, while the E stores 2.5GB and the K stores 3.75GB per disk. The J/K family also has a shorter seek time than the D/E family.

A string of DASD consists of a head of string device (designated by the letter "A"), and "dumb" devices (designated by the letter "B"). A head of string contains the intelligence needed to perform disk operations for itself and all B devices in the string. There is a maximum of three B devices per A device, and the strings must be all of one type (i.e., all D/E or all J/K type). There also exists a DASD which does not require a controller; this is designated by the letter "C".

Magnetic Tape Controllers and Devices

The evaluated line of tape devices is not as extensive as the DASD line. There are two controllers: the 3480-A11 and the 3480-A22, which control the 3480-B11 and the 3480-B22, respectively. The 3422-A01 acts as the head of string, while the 3422-B01s can be attached to fill out the string. The B11 drive is essentially a half-speed B22; the B22 transfers data at approximately 3 MB/sec. The 3480 devices are 8 track, 38,000BPI, 200 MB capacity cassettes, while the 3422s are the traditional 10.5" reel-to-reel tape drives.

Terminal Controllers and Terminals

The 3274 control units manage the transfer of data to and from terminals attached to them. 3274s are attached directly to channels of the host data processing system. The control units are customized at the site to support a selected configuration. The 3174 control units are newer versions of 3274 possessing similar characteristics.

Physically, the 3274 and the 3174 control units are small, floor-standing devices. Each one is capable of controlling of up to 32 locally attached display stations (terminals). The major differences between models are in the amount of memory the units possess and the attachment method. Only controllers utilizing the local attachment are evaluated. This includes 3274 Models 41A and 41D, and 3174 Model 1L. All these models connect to the system through a byte or a block multiplexer channel. All models partition their storage among the terminals they are controlling.

The control units are identified to the system via 2-byte control unit addresses. The system uses these addresses as I/O addresses. Terminals are identified to a control unit via 2-digit numbers corresponding to the terminal ports of the control unit.

The 3178 and the 3278 display stations are IBM's monochrome terminals. Both offer a variety of screen formats and feature many display characteristics. The 3179 and the 3279 display stations are the color equivalent of the 3178 and the 3278. Again, various models are offered. All the terminals attach to the control units (3174 and 3274) via terminal adaptors.

Printers

The IBM 3800 printing subsystem devices are high-speed, general purpose, electrophotographic printers. Control units are integral parts of these devices.

There are several models available. Model 1 is the basic model with the following characteristics: 52K byte page buffer, 20 character sets, vertical and horizontal spacing controls, forms overlay capability, job separation, and 31.8 inches per second printing rate (about 16,700 lines per minute), and 180 by 144 picture elements (pel) per square inch density. It is line-oriented and it attaches via the block or byte multiplexer channel.

Model 3 is an improvement over Model 1. It has these additional features: all points addressable mode (page orientation), 240 by 240 pels per square inch density, and an accumulator feature (for 768K byte to 1280K bytes of storage area for raster patterns). Model 3 attaches to the block multiplexer channel or to the data streaming channel adapter where transfer rates of 2.5M bytes per second are possible. Model 6 is a slower version of Model 3.

The IBM 3820 Page Printer uses non-impact laser print technology allowing all points addressability and high print quality. The 3820 uses the same pel density as the 3800 Model 3. It is attached to the system using either 3174 or 3274 controllers, and it may contain up to 4 megabytes of local storage.

The IBM 3262, 3626, 4245, and 4248 Line Printers are impact line printers with varying printing speeds and characteristics. They attach either to 3174 or 3274 control units, or to channels.

Interrupt handling

System 370-XA machines support six different types of machine interrupts: external, machine check, I/O, program, restart, and supervisor call (SVC). Whenever an interrupt is encountered, the current PSW is saved and a new PSW is loaded from a known memory location (the specific memory location is dependent upon the type of interrupt encountered). There are accompanying interruption codes for many of the interrupts, and in some cases there are also addresses for a parameter block.

The new PSW points to a first level interrupt handler (FLIH). The function of the handler depends upon the interrupt generated. The FLIH will save status associated with the job, then check any needed authorities (in the case of SVCs), and pass control to the appropriate control program after enabling interrupts. After the control routine has performed the action required by the interrupt, either control is returned to the routine that was executing when the interrupt occurred (non-preemptive), or the system dispatcher gets control (see page 41, "Dispatcher") and the highest priority ready unit of work is dispatched. All service request blocks are non-preemptive, and task control blocks are non-preemptive if the SVC associated with the control block is non-preemptive. These control blocks are further described on page 41, "Dispatcher".

External interrupts are generated by conditions from either inside or outside the system. Interrupts from outside the system may reach the CP only via hardware connections. Interrupts from inside the system are generated when clock events occur.

Examples of events causing external interrupts are the operator pushing the Interrupt key at the console, a CP in a multi-CP configuration losing power, and the Time of Day clock being in an error or non-operational state.

Machine check interrupts are generated when equipment malfunction is recognized by the system itself. Various codes indicate the severity and location of the component that caused the interrupt.

I/O interrupts are generated by I/O devices requesting service from the CP. In order for I/O interrupts to be serviced, the CP must have enabled interrupts for that device.

Program interrupts are generated when an execution of a program attempts to perform some action that the system does not allow. Examples of the types of actions that can cause program interrupts are attempts to access unauthorized memory locations, attempts to execute privileged instructions, and arithmetic errors (divide by zero, etc.).

Restart interrupts are generated when the operator requests a system restart. In a multi-CP environment a restart interrupt can also be initiated by one CP sending a SIGP instruction to another CP (see page 14, "PRIVILEGED and SEMI-PRIVILEGED INSTRUCTIONS").

Supervisor Call interrupts are generated when a SVC instruction is executed. Bits 8-15 of the SVC instruction contain the number (0-139) of the SVC that the program is requesting (see page 50, "SVCs in MVS/XA with RACF").

All external, I/O, and machine check interrupts are maskable, as are some program check interrupts. All others are non-maskable, and try to execute immediately. Priority for interrupts is: SVC, program check, repressible machine check (irrepressible machine checks execute immediately, as they usually signal a catastrophic failure), external, I/O, and restart.

SOFTWARE ARCHITECTURE

The system under evaluation contains a number of components, of which MVS is only a part. The components under evaluation are MVS/SP, JES2, DFP, ACF/VTAM, TSO/E and RACF. Specific information on these components can be found on page B-1, "Evaluated Software Components".

The MVS/XA operating system is a very large program. It is the main controller for the system under evaluation, and has many different modules. It is the controller for the rest of the products.

TSO/E provides users with an interactive command environment that interfaces directly with MVS/XA and its subsystems. Users can also submit batch jobs from the TSO/E session.

JES2 prepares jobs to be executed by MVS/XA by obtaining the resources necessary to execute the job. After completion of the job, JES2 releases those resources back to the system and prepares output, if any.

The Data Facility Product (DFP) handles all I/O processing for MVS/XA and its subsystems. DFP is invoked by all users, including MVS routines, to perform data set functions (e.g., open, close, allocate, copy, erase, etc.). It calls RACF to validate access authority and the I/O supervisor in MVS to create the channel programs to perform the physical reads and writes.

The Resource Access Control Facility (RACF) is a security add-on package for the MVS and MVS/XA operating systems. RACF provides identification and authentication, access control, and accountability mechanisms.

ACF/VTAM is the Virtual Telecommunications Access Method that is used to communicate with the host hardware. This runs on the hardware, and is necessary for TSO/E to become active over remote links.

These six products are described in detail in the following sections.

MVS/XA STRUCTURE

The MVS/XA Operating System contains many large modules. The following section describes the important and security relevant modules, as well as important data structures and facilities. As the name implies, the key to the system is many different virtual address spaces. The following description of address spaces also introduces some very important functional modules.

Address Spaces

Conceptually, an MVS/XA address space consists of two gigabytes of virtual storage. An MVS/XA address space contains the system prefixed save area, private areas, and common areas. Each user has an entire address space and thus has access to all three kinds of areas. MVS/XA effectively isolates one address space from another by means of segment and page tables previously described (see page 11, "DYNAMIC ADDRESS TRANSLATION"). Users can share programs and data areas through the common areas of the address space. Since MVS/XA has been made compatible with MVS/370, the layout of the virtual address space is somewhat peculiar (see figure 2).

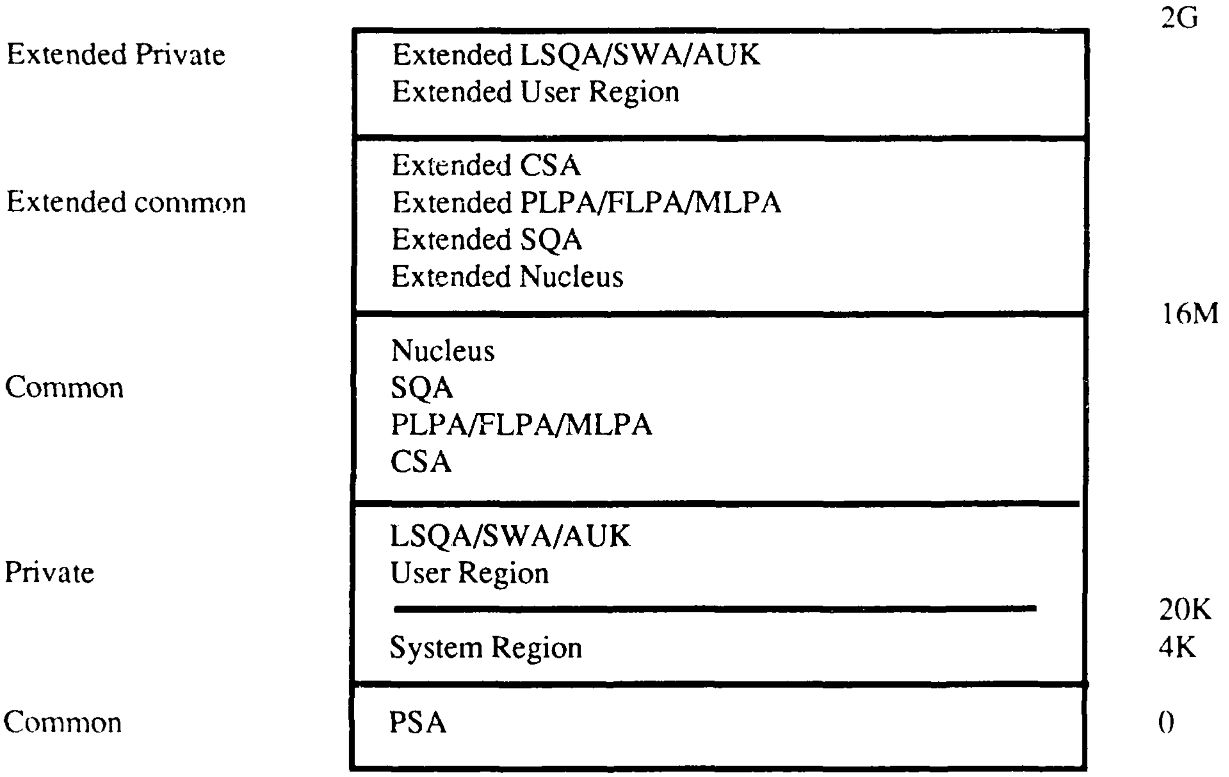


Figure 2

The virtual address space is, for convenience, divided into many different areas of storage called subpools. Pages in a given subpool have several common characteristics, including the location of the subpool in common, private, extended common, or extended private storage, the protect key that the subpool can be accessed with, whether the data in the subpool is fetch protected, when and how the subpool is freed, and whether the subpool is pageable or fixed in real storage. Subpools allow users to free data at once, even though it may have been allocated at different times earlier in the job. There are 255 subpools: subpools 0-127 are allocated for use by general users; subpool 128 is for compatibility with OS/VS1 programs; subpools 129-225 are undefined; the rest are allocated for system storage (SQA, LSQA, ELSQA, etc.).

The Prefixed Save Area (PSA) contains critical information about both the MVS/XA operating system and the processor(s). It includes fixed storage locations for interrupt handling, register save areas for system routines, and pointers to critical control blocks as described earlier.

The private area contains modules and data not shared by other address spaces. It consists of five sections: system region, user region, LSQA, SWA, and AUK. The system region is used by system functions performing work for an address space. These system functions run under the region control task (RCT), which is the highest priority task in each address space and plays a key role when an address space must be swapped in or out.

The user region and the extended user region are the sections of the private area in which user programs run. There are two types of user regions: virtual (V=V) and real (V=R). Virtual user regions are pageable and swappable. These are the regions most often used in a timesharing system. Real regions occur only below the 16 megabyte line, and are non-pageable and non-swapable. Although DAT is used, the virtual address always corresponds to the real address.

The system administrator, at sysgen time, determines what the size of the real region shall be. After IPL, whenever a user specifies ADDRSPC= on a job step card (see page 55, "Job Control Language, Jobs, and Tasks"), one of three actions can occur. If the requested space is greater than the space specified at sysgen, then the user job step will not run. If the requested space is not available (i.e., other users have already allocated it), the job step will wait. If space does not become available when the job step timeout has expired, the job step again will fail to run. If there is enough space, then the user is assigneda key from 9-15, whichever the next sequential key number is that is not being used, and will begin to execute in that region. If all keys are being used, then the job step will wait for a key. If one does not become available before the job step timeout has expired, the job step will be terminated. When users are not executing in this region, the system uses it as a fast paging area. When it is required, the pages are migrated to the expanded store.

The authorized user key (AUK) area and the extended authorized user key area of the private region contain system data relating to a specific user. Protected user control blocks reside in this area. AUK also contains data for the LNKLST lookaside (LLA), which is an in-storage directory of all of the modules that are in SYS1.LINKLIB. The LLA address space provides a cross-memory search routine used to speed up searches of SYS1.LINKLIB by other components of the system. The scheduler work area (SWA) and the extended scheduler work area contain the control blocks that exist from job step initiation to job step termination. These contain the internal (interpreted) form of the Job Control Language (JCL) (see page 55, "Job Control Language, Jobs, and Tasks") statements that accompany a job. The local system queue area (LSQA) and the extended local system queue area contain tables and queues that are unique to a particular address space such as the user's segment table and private area page tables. LSQA also contains all the control blocks that the RCT requires. LSQA is swappable but not pageable.

The common area holds system information, such as program code, control blocks, tables, and data areas. The common save area (CSA) and the extended common save area are addressable by all active programs and they are are used by all swapped-in users for inter-address space communication. CSA contains some fixed and some pageable system and user data areas. The pageable link pack area (PLPA) and the extended pageable link pack area contain MVS/XA control program functions (SVC routines), access methods, other read-only system programs, and selected user programs. PLPA is pageable but no physical page-outs occur since PLPA provides read-only modules. The fixed link pack area (FLPA) and the extended fixed link pack area are fixed in storage. FLPA consists of modules which could be in PLPA but because of fast response requirements are fixed instead.

The modified link pack area (MLPA) and the extended modified link pack area can be used for reentrant modules from selected system or user libraries. MLPA exists for the duration of the active MVS/XA system but it is not saved from one MVS/XA start-up to another. The system queue area (SQA) and the extended system queue area contain tables and queues that relate to the entire system. For example, the page tables that define the system area and the common area reside in SQA.

The nucleus and the extended nucleus hold the resident part of the MVS/XA control program. In addition they contain the page frame table entries, DEBs for the system libraries, recovery management support routines, and unit control blocks (UCBs) for the I/O devices. While most of the nucleus executes with the DAT enabled, certain recovery processing routines within the nucleus execute with the DAT disabled. The DAT-on nucleus itself is made up of four types of modules: read-write modules using 31-bit addressing, read-only modules using 31-bit addressing, read-only modules using 24-bit addressing, and read-write modules using 24-bit addressing.

Cross Memory Services

As mentioned above, MVS/XA provides each user with a unique address space and maintains the distinction between the code and data be'onging to each address space. MVS/XA also includes cross m nory services that permit a single user to access other address spaces when necessary.

Cross memory allows programs to pass control to programs in other address spaces and to move data from one address space to another. Because a program using cross memory capabilities can directly access programs and data in the private area of another address space, cross memory can reduce the amount of common area needed in the virtual address spaces in the system.

Cross memory services execute out of PC/AUTH address space creating and managing the data structures that support the program call (PC) instruction and allow control of the cross memory authorization structure. These services are used by supervisor state or PSW key 0-7 callers, usually subsystems, to set up the environment for controlling cross memory access to programs and data. This restriction excludes non-authorized users from directly utilizing these services.

PC/AUTH services are used by MVS/XA itself. Whenever an address space is created (see also page 36, "Address Space Creation"), PC/AUTH initializes its address space second table entry (ASTE) and chains the new address space to the system linkage table (SLT) and the system authorization table (SAT). The new address space is unauthorized to issue the program transfer (PT) instruction to another address space or the set secondary address global PC services that are available to all address spaces via the SLT.

When a task or an address space terminates, the PC/AUTH resource manager gets control. If a cross memory resource owning task or address space is terminating, PC/AUTH-related resources are recovered.

Finally, PC/AUTH provides services that allow authorized programs (supervisor state or key 0-7) to build and manipulate entry tables and linkage tables used for housekeeping while cross memory functions are performed.

Storage Managers

There are three types of storage in the MVS/XA system, each of which has its own storage manager: the real storage manager (RSM), the auxiliary storage manager (ASM), and the virtual storage manager (VSM). These three components are discussed in this section

Real Storage Manager

Real storage refers to the main memory of the system, which can vary according to the model of the processor that is running MVS/XA. RSM manages the real storage, and also the expanded storage on the system. MVS/XA uses the expanded storage region as a solid-state paging device, which conceptually lies between real storage and auxiliary storage.

Functions of RSM include handling segment and page faults, providing paging services such as fixing and freeing pages in main memory (i.e., making the pages unpageable and pageable, respectively), paging storage out (in conjunction with ASM), and releasing and loading pages. Other services provided by RSM are virtual I/O and virtual fetch services, real storage reconfiguration, V=R allocation, address space creation, swapping, storage and key error handling, page migration, and virtual data access services.

Real storage is divided into 4K blocks called frames. In addition to managing which frames are currently in the configuration of the system, RSM controls which frames are occupied by the pages themselves. If an error occurs when accessing a frame, one of three actions will be performed. If the error is an uncorrectable storage error, the frame is placed offline. If the error is an uncorrectable key error, the key is refreshed; if this is unsuccessful, the frame is take offline. For correctable errors, an appropriate service request block (see page 41, "Dispatcher") is scheduled.

Page stealing is a process which is managed by RSM. The system resources manager (see page 41, "System Management") tells RSM which pages to steal by providing information on the address spaces which are candidates for having their pages stolen. This is done by selecting the least recently used page frame. RSM will then take that page frame for use in a different address space, thus reducing the working set for the address space the page was stolen from.

Page migration refers to the movement of pages from extended storage to auxiliary storage. This will happen either because there is a shortage of available frames in extended storage, or because the extended store is being configured off-line. In the first case, RSM invokes the purge migration and LRU migration routines. In the second case, RSM invokes the reconfiguration migration routines.

RSM also manages the swapping of address spaces into and out of real storage. If a machine has extended storage, there are four swap functions that RSM can perform: swapping to and from auxiliary storage, and swapping to and from extended storage. SRM decides which address spaces are to be swapped, and tells that address space's region control task (see page 38, "MVS/XA Initiation"). RCT then calls RSM to make the actual swap. An address space on extended storage can also be migrated (by RSM) to auxiliary storage.

Virtual Storage Manager

Since virtual storage, as previously described, is used by every user on the system, a manager is needed to supervise virtual storage operations. The VSM is responsible for managing the subpools, defined previously, which includes keeping track of which subpools are allocated and how much free space is in each allocated subpool. The VSM is also responsible for servicing ten external macro instructions and for supporting virtual storage operations during system initialization. Two important macros are GETMAIN and FREEMAIN. These two macros are used to allocate and deallocate virtual storage for a process. Other macro functions include obtaining information about which areas of storage are allocated and which are free, verifying which locations are allocated to certain virtual storage areas (such as LSQA), and operations for manipulating cells, which are small areas of virtual storage. The VSM also is responsible for obtaining the storage protection key from the unit of work, and passing it to RSM for checks against the key on the page frame.

Auxiliary Storage Manager

The ASM manages storage that is not in the main memory; for example, disk storage as it relates to processes is managed by ASM. The specific types of data that are managed by ASM include the content of page data sets, swap data sets, and VIO data sets. The ASM units are called slots, as opposed to pages for virtual memory, and frames for real memory. The ASM primarily interacts with RSM for paging operations, and for swapping address spaces to and from main memory.

I/O Onerations

I/O operations actually begin when a user program issues an OPEN instruction. The user program provides a data control block (DCB) to the OPEN macro. The macro will fill the DCB with information such as the device and data set information (from the job file control block), specific information from the data set control block, and addresses for the access method routines (see page 65, "Access Methods") to be used for this data set.

The OPEN routine also builds a data extent block (DEB), which contains pointers to the DCB, a structure called the unit control block (UCB), and appendage routines (user supplied exits, for example). Appendages can gain control anywhere in the I/O cycle, possibly changing the channel program. Appendages can only be established by authorized routines from authorized libraries (see page 54, "Authorized Programs") and can only be loaded for authorized users.

Thereafter, whenever the user program issues an I/O instruction (e.g., GET, PUT, READ, WRITE) the access method routines get control. These access method routines build an I/O block (IOB), an event control block (ECB), and the necessary channel program. The IOB contains pointers to the DCB, the ECB and the channel program. The ECB contains information about the status of the I/O so that the user program can access its contents to see the results. The access method then issues an Execute Channel Program (EXCP) instruction to pass control to the EXCP processor.

If a user wishes, the user may choose not to use any of the standard access methods but rather communicate directly with the EXCP processor. This of course means that the user must fill in all of the information that the EXCP processor expects. Before proceeding, the EXCP processor will perform address checks by obtaining valid address ranges from the DEB. The DEB is created by the OPEN macro, which every user must call in making an access (even those writing their own access methods); thus the valid address ranges can not be specified by the user.

EXCP is a driver for the I/O Supervisor (IOS), which communicates directly with the channel subsystem to perform the I/O. EXCP has three parts: the front end, which prepares the request; exit processing, which monitors and handles interrupts; and the back end, which handles clean up functions and status returns. The front end of the EXCP processor first creates an I/O Supervisor Block (IOSB). The IOSB contains the address of the UCB, and the address of the channel program that is translated by EXCP.

The channel program is copied from the user space to system space so that it can not be tampered with, and then translated (i.e., put into a form that the IOS can understand), converting the virtual storage addresses to real storage addresses. The front end also fixes the I/O buffers that are used to hold the incoming or outgoing data because the channel subsystem does not know about virtual addresses. Another function of the front end involves prepending a track or cylinder interrupt routine to every channel program. The purpose of this prepended portion is to ensure that when a new track or cylinder is beginning to be read, the user is authorized to read that track or cylinder. Before a new track or cylinder can be read, the prepended portion of the channel program causes an interrupt; this allows the EXCP exit processor to get control. The exit processor will check the DEB to make sure that the new track or cylinder is readable (writable) by the user, and then either fail or proceed, depending on the outcome of the decision. The EXCP processor then issues a STARTIO instruction to pass control on to the I/O Supervisor.

When EXCP issues the STARTIO macro, control returns to either the access method or the user program, depending on the type of macro used to initiate the I/O. The access method (or the user program) will then wait on the ECB, which means that it will wait until status is posted to the ECB before continuing.

The IOS chooses the subchannel to be used and creates an Operation Request Block (ORB). The ORB contains the necessary information for the channel subsystem to perform the I/O request. IOS then invokes the channel subsystem by issuing the Start Subchannel (SSCH) instruction with the address of the ORB being passed as an argument.

The channel subsystem fetches the first Channel Command Word (CCW) to ensure it passes certain validity checks. There are two types of CCWs: format 0 and format 1. Although both formats contain the same types of information (command code, flags, byte count, and data address), format 1 fields are arranged differently in order to accommodate 31-bit addresses. Assuming the first CCW passes the validity tests, a channel path is established. Command codes from the CCWs are then sent down this path to the device, initiate the physical I/O. If a path cannot be found, the operation remains pending until one opens up.

The channel subsystem supervises the execution of the channel program, transfers the data, updates the relevant control blocks and posts I/O interrupts when necessary. EXCP monitors these interrupts, and takes the appropriate actions. For instance, EXCP could pass control to a certain appendage specified in the DEB. After the channel subsystem has completed the request, IOS regains control. It will examine the status, found in the IOSB, and either initiate error processing, or return control to the EXCP back end.

The EXCP back end posts the results of the I/O in the ECB. The access method or user program can test the contents of the ECB to determine the outcome of the I/O. The back end also "unfixes" the storage buffers used for this I/O.

Virtual Input/Output

Virtual Input/Output (VIO) is a method of performing I/O for temporary data sets that eliminates the time-consuming transfer of data using the channel subsystem. Temporary data sets are the only type of data set than can use VIO, and only for the duration of the job which creates them. VIO uses the system paging routines for data transfer

VIO moves data from the channel program's data transfer buffer to an area in the user's address space known as a window. The size of a window is commensurate with the size of a track on the device specified on the data set definition statement.

When the system is initialized, certain I/O unit names are assigned for VIO. Thereafter, when VIO is to be used, the user can specify the unit name, and the system will create a temporary data set that has a system-generated name. When EXCP is translating the channel program, it will invoke VIO instead of IOS for performing the data transfer.

When transfering data, VIO uses a long move instruction to move the data between the window and the buffer. If the window becomes "full" (i.e., a track boundary is crossed), VIO will write the contents of the window to a page data set, and then sever the connection between the two. Thus when the write begins again (i.e., the second track is being written), a page fault will occur, and blank page frames will be associated with the window.

When reading data back in, VIO locates the necessary pages in auxiliary storage (if the pages are not currently in the window), and then sets the page table entries of the window to point to those pages that contain the desired data. The page table entry invalid bits are set, causing the resulting page fault to bring in the desired pages. Since RSM (see page 31, "Real Storage Manager") tries to keep these pages in real memory as long as possible, there is a good chance that no physical I/O will be done.

Address Space Creation

User address space creation revolves around three address spaces: master scheduler, user, and JES. A batch job will not run in an address space created especially for it, but will instead run in an initiator's address space (described later). Initially, after receiving a START, MOUNT, or LOGON command, the master scheduler invokes the address space creation routine which assigns an address space identifier, creates control blocks, and requests concurrence from the system resources manager (SRM). The master scheduler then either releases control blocks or invokes VSM to assign virtual storage, set up addressability, build LSQA, and create RCT control blocks. At this point the execution resumes in a user's address space.

The region control task builds control blocks and invokes the started task control (STC). The STC determines which command is being processed and builds in-storage JCL text for the job. The execution now transfers to the JES address space where JES reads the job, scans the JCL and writes it to spool, invokes the converter to transform the job to internal text, queues the job, assigns a job identifier and passes it to the initiator. The initiator routine is a part of the STC within a user's address space. It requests JES to prepare the job for execution. JES--in its address space--invokes the interpreter to build control blocks from internal text, and passes control back to the initiator in the user's address space. The initiator then invokes the allocation routines and finally attaches the appropriate program/processor. The START command results in the execution of a specified program, the MOUNT command invokes the MOUNT command processor, and the LOGON command activates the terminal monitor program (TMP). TMP controls the interchange of user commands with TSO/E.

Initiators also are used by JES to run job steps that JES has prepared for execution. The initiator will request work from JES according to the job class (see page 55, "Job Control Language, Jobs, and Tasks"), and the job that JES passes to the initiator will then run in the initiator's address space. The initiator is responsible for cleaning up all control blocks associated with the jobs it runs. An initiator can be assigned one or many job classes.

The region control task (RCT), as mentioned above, is the highest priority task in an address space. Its functions are: preparing an address space to be swapped out; preparing an address space for execution after a swap-in; and ensuring proper scheduling of a user attention exit. When SRM determines that an address space should be swapped out, the RCT sets all tasks under the RCT as non-dispatchable, purges all its I/O requests, copies the saved functional recovery routine stacks from the SQA to the LSQA, breaks active addressing binds from other address spaces, and finally calls the RSM swap-out routine to initiate the swap-out.

When the address space is swapped in, the RCT common processing invokes the restore routine which prepares the address space, reschedules purged I/O requests, and sets all tasks under the RCT as dispatchable. When a user requests an attention exit, RCT routines ensure that it is properly scheduled and executed.

The started task control (STC) routines oversee the initialization of system component address spaces and the processing of START, MOUNT, and LOGON commands. The initialization could be for either a limited or a full function address space. A limited function address space can not allocate data sets, read JCL procedures from the system procedure library, allocate a SYSOUT file, or use system services running in cross memory mode. A full function address space does not have these restrictions. All system component address spaces are limited function address spaces except the dumping services address space (DUMPSRV) and the system management facility (SMF) address space.

The STC routines perform five major functions: obtain the region in which STC will run; determine which command was specified; build internal JCL text for the command task; build the control blocks required for initiator/terminator processing; and free those control blocks after the initiator/terminator terminates the command task.

The purpose of the initiator/terminator is to make all the necessary preparations for the execution of a job step or job task. In order to accomplish this, the initiator performs the following functions: obtains storage for a task; initializes the control blocks for a task; assigns properties to a task; oversees the allocation of data sets and devices for a task; opens any required catalogs and libraries for a task; and attaches (spawns) the task.

When a task has completed execution, the terminator performs the following functions: deletes the control blocks no longer needed, deletes the RACF accessor environment, oversees the freeing of data sets and devices used by the task, and detaches (kills) the task. When an entire job is complete, the initiator clears and deletes the control blocks and data areas used as well as the storage space occupied.

The initiator provides the above functions in four situations: completing master scheduler initialization; starting a subsystem (such as JES); processing a START, MOUNT, or LOGON command; and initiating a normal job. In the first three situations, the initiator is used as a subroutine to initiate a single job. When that job is completed, the initiator subroutine returns to its caller. In the last case, the initiator is a task created as a result of a START command, i.e. the initiator is a started task. This initiator can, in turn, attach another task. When that attached task completes, the initiator requests another job from the job entry subsystem. JES returns to the initiator with either another job or an indicator to stop processing. In a typical configuration, there may be several such initiators each executing a batch job.

MVS/XA Initiation

The initialization process consists of loading the nucleus, initializing system resources and resource managers, initializing system component address spaces, and initializing the primary Job Entry Subsystem (see page 55, "JOB ENTRY SUBSYSTEM 2"). The process is divided into three phases: initial program load (IPL), nucleus initialization program (NIP), and master scheduler initialization.

System initialization begins when the system operator initializes the hardware. This is done by invoking the initial microprogram load (IML) to start the processors, by mounting the necessary disk and tape volumes, and finally by requesting the LOAD function. This function activates the IPL control program which in turn utilizes IPL resource initialization modules (IRIMs).

The IPL program clears real storage, prepares an environment in which the IRIMs can execute, controls the loading and deleting of the IRIMs, and provides basic service routines for this phase of initialization.

The first IRIM loads the nucleus while another builds the DAT-off to DAT-on linkage table used to establish addressability between entries in the DAT-off nucleus in real storage and entries in the DAT-on nucleus in virtual storage. Other IRIMs initialize or reserve storage for many system component control blocks, work areas, and programs. The IRIMs also begin to initialize the private area of the master scheduler address space, which is the first address space to be created. The VSM IRIM reserves storage in SQA for the system tables and queues. It also sets up extended LSQA with tables and queues to be used by the master scheduler. A RSM IRIM initializes a segment table whose entries are the addresses of page tables for the common area of virtual storage. This common segment becomes a part of the master scheduler address space segment table. Another RSM IRIM initializes the tables that identify how the frames of real storage are assigned. This table resides in the read-write extended nucleus. Yet another IRIM builds the PSA.

NIP processing is the second phase of the initialization process. NIP utilizes resource initialization modules (RIMs) in establishing the master scheduler address space. This process is completed when the common segment table is copied from the master scheduler's private area into SQA for all address spaces to use. VSM and ASM RIMs allocate virtual storage in the common area for CSA, SQA, and LPA. IOS RIMs perform device initialization by building the unit control blocks (UCBs) and the installed channel path table. Building UCBs requires initializing the channel subsystem, testing the availability of a device, testing the accessibility of a device, and then checking for duplicate volumes. The master catalog, used to locate cataloged data sets and other catalogs, is then initialized, as is ASM. Page and swap data sets are also opened and initialized.

During this phase, the program call/authorization (PC/AUTH), system trace (TRACE), global resource serialization (GRS), and DUMPSRV address spaces are initialized. The PC/AUTH routines initialize all the cross-memory tables needed to establish communication with other address spaces. Other system component address spaces use PC/AUTH services to create and initialize their own cross-memory tables. TRACE provides for tracing of system events, installation-defined events, and component events (i.e., certain events that occur in component address spaces). GRS serializes the use of local and global serially reusable resources (see page 49, "Global Resource Serialization").

Initializing the master scheduler is the final phase of the system initialization process. In this phase routines required by system-initiated cancel, SWA management, and resource management are loaded. Control blocks needed to invoke the initiator are created and initialized. Then the master scheduler base initialization routine initializes the subsystem interface, the communications task, and initial TSO/E addresses. It performs master trace initialization, sets the time-of-day clocks, and it attaches the initiator to start the master scheduler.

The initiator allocates the required data sets and internal reader data sets, and invokes RACINIT to establish the security environment. These will be later used to pass JCL from system routines to JES. Finally, the initiator attaches master scheduler region initialization as the job step task thereby activating the master scheduler. The master scheduler accepts system commands and activates their processing. The START JES command starts the initialization of the job entry subsystem.

While initializing the JES, the master scheduler creates an address space for JES. This is accomplished via the address space create routine which builds LSQA in the private area and initializes segment tables and page tables to represent the new address space. Then the routine builds task control blocks for a region control task (RCT) and places the address space control block (ASCB) on the dispatching queue. When the JES address space becomes active, the first task dispatched is the RCT. After the RCT is initialized, it attaches the STC to initiate JES. The STC does so by building job scheduler control blocks in the SWA while the initiator allocates the required data sets. Finally, the initiator attaches the primary JES and MVS/XA begins accepting jobs.

Before TSO/E logons can be accepted, VTAM and TCAS must be initialized by the operator via a START command. Both VTAM and TCAS operate in their own address spaces. After these two address spaces are initialized, a TSO/E logon can be accepted by the system. This completes the system initialization process. During this phase five new address spaces have been created: CONSOLE, ALLOCAS, SMF (see page 107, "System Management Facilities"), JES2 (see page 55, "JOB ENTRY SUBSYSTEM 2"), and LNKLST (the LNKLIST lookaside address space; see page 28, "Address Spaces"). From this point on, user address spaces may be created.

Although most of the allocation of control blocks is done in the user's own address space, the ALLOCAS address space contains the control blocks used by the unit allocation status recording module. In addition, the allocation address space initialization routine and the display allocation tables manager both execute in the ALLOCAS address space.

The CONSOLE address space contains the communications tasks. These tasks are primarily used for communications between a user and a system console or a TSO/E monitoring device. The Write to Operator, Write to Operator with Reply, and Delete Operator Message macros are used to perform the communication. The issuer of the macro can specify the console to which the message is directed.

Consoles

There are two types of consoles on a 3090 system: consoles which are attached to the PCE directly, and consoles which are attached to the 3090 machine, usually through a control unit such as the 3174. This section describes both of these types of consoles, and their function in maintaining and operating the system.

The consoles that attach to the 3090 are called the operator consoles. There is one master operator console on the system, and optionally up to 31 other operator consoles. There is also a provision for switching the master console from one device to a back-up device (defined at sysgen, as all the other consoles are) by causing a hardware interrupt at the operator control panel. The master console is the operator console which has the lowest port address. All operator consoles execute in the CONSOLE address space. These consoles control MVS/XA functions, and display messages relating to MVS/XA performance, user requests, etc. The subchannels that the consoles are attached to are defined in a data set. This data set is read at IPL, and the changes to the consoles are put into effect at that time. In order to add a new console, the data set must be edited, and the system re-IPLed.

Consoles which attach directly to the PCE are different from operator consoles. It is important to distinguish between the displays and the consoles. The displays are the physical terminals (CRTs) which attach to the PCE. There are four display ports--two service displays and two system displays. There are many logical consoles, each of which can be associated with the same display. Logical consoles are the system, service, program mode, system monitor, service monitor, data bank access, and remote consoles. Physical displays can only be assigned one console at a time, and some consoles can only be assigned to one display at a time. The two most important consoles are the system and the service consoles.

A minimum console configuration for a 3090 machine is one system display for each SCE, and a service display. The system console has two access levels. Level 2 (the lowest access level) gives the operator access to configuration, control, and monitoring functions. Level 1 gives access to level 2 functions, well as recovery functions.

The service console also has two levels. Level 2 is used for diagnostics, while level 1 is used for PCE address and channel manipulation, as well as level 2 functions.

The service displays can be located a maximum of 10 feet from the 3090 (co-axial cable hookup), and the system displays can be located up to 1500 feet away. They are considered part of the physical hardware of the system, and are treated as such.

System Management

Managing the work in a system such as MVS/XA is a complex task. Many functional modules make up the MVS/XA system. The following section will describe the process dispatcher, the System Resources Manager (SRM), and serialization methods. Although these modules do not entirely describe the system, they are a good overview of the important, security and integrity relevant modules and functions. In addition to describing the functions of the modules, relevant data structures will also be described.

Dispatcher

The dispatcher for MVS/XA is responsible for removing a "unit of work" from a queue and handing it to a processor for execution. Units of work, when ready to execute, are called dispatchable units. The dispatcher will dispatch the highest priority unit of work that is available. Units of work are represented in three ways on the system: as special exits, as Service Request Blocks (SRB), and as Task Control Blocks (TCB). Although there are many queues of different types of SRBs, there is only one queue of TCBs per address space.

Special exits are extremely high priority service routines that must run as soon as they become ready. These are branched to directly by the dispatcher, instead of being handled in the normal manner. An example may be the exit that is invoked when a processor crashes, and another processor on the same system is attempting alternate CP recovery on the failed CP.

A non-preemptive unit of work is one which can be interrupted, but must receive control when the interrupt is done. A pre-emptive unit of work will not receive control; in this case, the dispatcher is invoked and will determine the next available unit of work. SRBs are non-preemptive requests for service from or for a particular address space. SRBs can only be created by a key 0 program running in the supervisor state. SRBs can run either in the address space in which they were created, or in another address space.

TCBs are control blocks for the "normal" work done on the system, e.g., user tasks, most system tasks, and started jobs. The ATTACH macro allocates the space needed for a TCB, chains this storage to the list of TCBs currently in the address space (making it a "subtask" of the TCB preceding it in the chain), and calls tne program management routines to find the program that the TCB will represent (this is the executable code). As opposed to SRBs, TCBs are preemptive, except for those that are controlling non-preemptive SVCs.

All tasks associated with an address space are in a list of TCBs (see figure 3). A header block, called the Address Space Control Block (ASCB), resides in the SQA. The ASCB points to the Address Space Control Block Extension (ASXB), which is located in the LSQA. The ASXB is what points to the head of the chain of TCBs. The first TCB in every address space is the RCT. The three types of address spaces that are associated in some manner with users of the system are the batch address spaces, the operator address spaces, and the TSO/E user address spaces.

In each of these address spaces, the RCT points to the DUMP TCB, which in turn points to the started task control TCB. At this point, different TCBs are chained. For batch address spaces, the STC TCB points to an initiator, which in turn points to batch TCBs (representing units of work requested by the user who started the batch job). For operator address spaces, the STC TCB points to started tasks. For TSO/E users, the STC TCB points to a terminal monitor program; TCBs representing jobs run by the TSO/E user are then pointed to by the terminal monitor program TCB

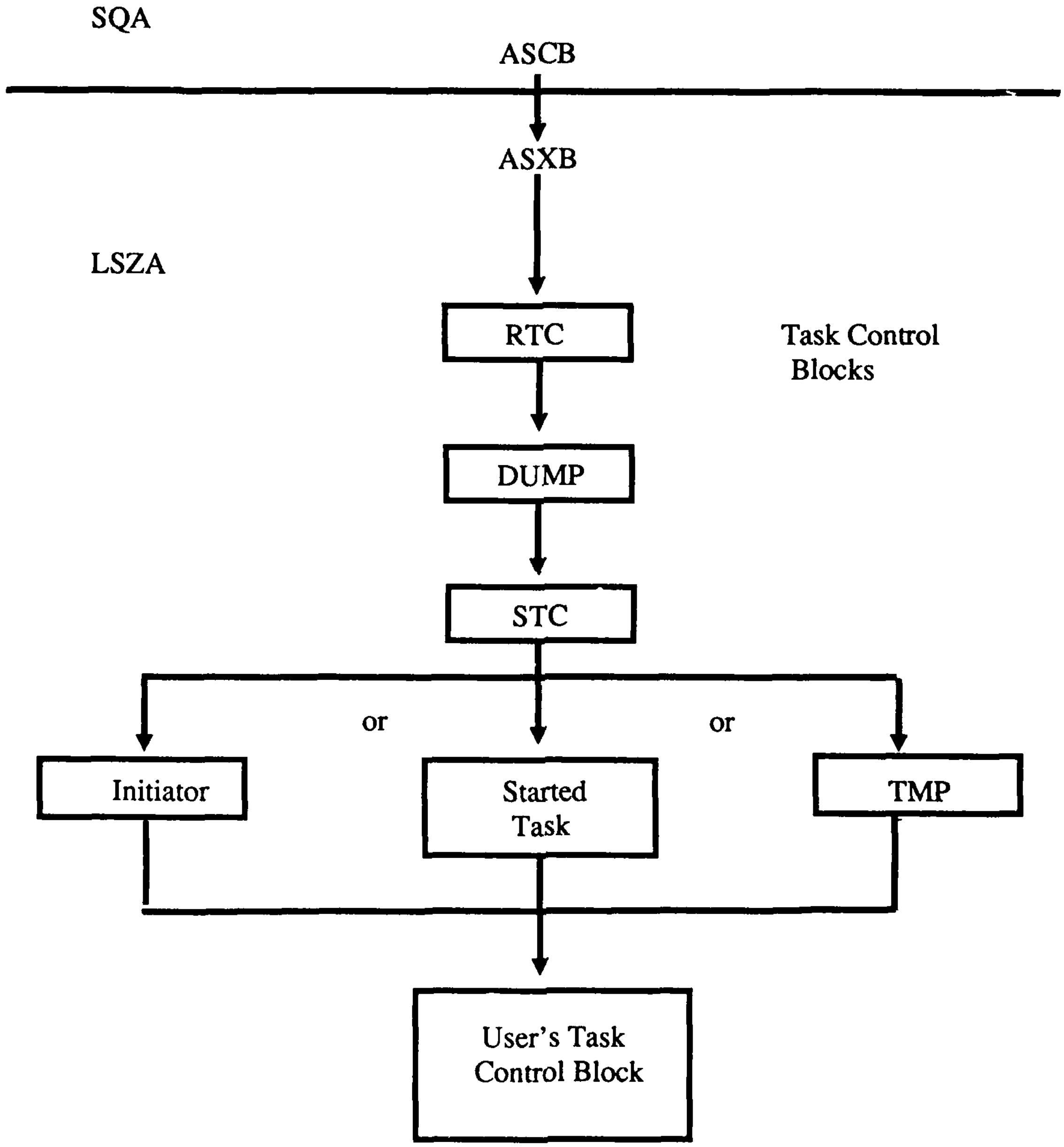

SQA
ASCB
ASXB
LSZA
RTC
Task Control Blocks
DUMP
STC
or
or
Initiator
Started Task
TMP
User's Task Control Block

Figure 3.

The following list gives the dispatching priorities, that is, the order in which the dispatcher looks for available work.

1. Special exits

2. Global priority SRBs on the global service priority list (GSPL)

3. Global priority SRBs on the global service management queue

   (Note that the global and local service management queue are merely staging areas for the global and local service priority lists. The SRBs are moved to the GSPL and LSPL before being dispatched.)

4. SRBs in the single local service management queue (LSMQ)

   (This is a single queue whose elements are waiting to be dispatched to a particular address space. They will be put on the LSMQ for the target address space from this queue.)

If no dispatchable units of work have been encountered, the dispatcher will now look for ready units of work on the true ready queue (described below).

5. Local priority SRBs on the local service priority list (LSPL)

6. Local priority SRBs on the local service management queue

7. Lock holders on this address space (see serialization discussion)

8. Highest priority TCB on the TCB chain for this address space

If there is no work in this address space, then the next address space on the true ready queue is examined in the same manner. If none of the address spaces on the true ready queue have dispatchable units of work, then the wait task is dispatched.

The true ready queue consists of the ASCBs of the address spaces which claim to have ready work. Changes while in the queue may invalidate this claim. The queue header is also an ASCB, and the last ASCB on the queue points back to the header. An ASCB is deleted from the queue when the task that was just dispatched was the last dispatchable task for that ASCB, or when no ready units of work are found on that ASCB.

System Resources Manager

SRM is responsible for controlling and monitoring system resources in order to optimize performance and gain maximum utilization of those resources. It is responsible for providing swapping and page stealing decisions to RSM, dispatching priorities, memory utilization analysis, and inhibiting or calling for increased creation of address spaces. SRM is divided into three parts: SRM control, the workload manager, and the resource manager.

SRM control has two functions: scheduling other SRM routines to run as needed, and making swapping decisions. Conceptually, there are four different types of swapping that SRM control dictates. RSM, which actually swaps the address spaces (with the help of ASM), is invoked by SRM control, and does not know which type of swap decision that SRM made. The four types are:

> Unilateral swap-out: SRM will begin to swap out address spaces when there are too many (as determined by a pre-set "high" level) in memory at the same time. It will continue to swap out address spaces until an acceptable level is reached. Since each address space consumes resources at different rates, the level is a measure of programming throughput, rather than a hard-coded number of address spaces.
>
> ENQ exchange: ENQ is one of the system serialization mechanisms (see page 47, "Serializing Resources"). If a swapped out user is ENQed on a resource that is needed by another user, then the swapped out user will be swapped back in.
>
> Exchange swap: If SRM determines that a user is using resources at too great a rate, then it will swap out that user, and swap in a user that does not use as many resources until the load on the resources is reduced.
>
> Unilateral swap-in: SRM will swap in address spaces if it determines that the system resources are being under-utilized.

Swapping decisions made by SRM control are based on input from the other two parts of SRM.

The workload manager section of SRM has three functions: monitoring the rates at which address spaces are consuming resources, providing swapping recommendations to SRM control as requested, and collecting data on resource utilization for measurement tools (such as the Resource Management Facility).

The resource manager section of SRM monitors and manages four areas: storage management, I/O management, processor management, and resource monitoring.

The storage manager takes action when one of the following shortages is detected: free page frames in real storage, pageable frames in real storage, available slots in auxiliary storage, and space on the SQA (which means the SQA may expand into the CSA).

If a shortage of free page frames is detected, SRM initiates page stealing until the shortage is alleviated. If a shortage of space on the SQA is detected, SRM disables new address space creation until the SQA has sufficient storage restored as a result of old address spaces terminating. The actions taken on detection of shortages of auxiliary slots and pageable frames are the same. SRM will reduce the workload by disallowing the creation of new address spaces and delaying any newly created address spaces from executing. SRM also swaps out the top user of the scarce resource, and notifies the operator of the identity of that user.

The I/O manager makes recommendations to SRM for swapping decisions. It also determines which device to allocate when more than one of the type requested by the user is available.

The processor manager controls the dispatching priorities of address spaces, prevents the swap-out of users ENQed on a resource that other users are waiting for, and it also makes swapping recommendations to SRM control.

The resource monitoring function makes recommendations for adjusting the number of address spaces currently in memory based on the rate at which those address spaces are consuming resources.

System Authorization Facility

SAF provides an interface to RACF (see page 76, "RESOURCE ACCESS CONTROL FACILITY") and/or to a user supplied processing routine by using a system service called the MVS router. The MVS router is the focal point and common system interface for all products providing resource control. The MVS router is always present in the MVS operating system. The MVS router provides an exit that will invoke either RACF or an installation written security processing routine.

The resource managing subsystems (e.g., JES, DFP) invoke the SAF MVS router by issuing the RACROUTE macro instruction. A parameter telling which RACF macro to process and the associated parameters needed are passed via RACROUTE to the MVS router. The MVS router then calls the MVS router exit, which returns to the MVS router with a return code indicating whether or not to invoke RACF. After invoking RACF, the MVS router converts the RACF return and reason codes and passes them to the caller via RACROUTE. The RACROUTE return code indicates that either the requested security function was completed successfully, the requested security function was not processed (possibly because RACF is not active), or the requested security function was processed and failed.

The RACROUTE macro instruction is used to access the RACF functions provided by: RACDEF, RACINIT, RACXTRT, RACLIST, RACHECK, and FRACHECK. Issuers of the RACROUTE macro instruction enter the MVS router in the same key and state as the RACROUTE issuer. Authorized programs may issue the RACF functions directly but RACROUTE is the preferred method.

Serializing Resources

MVS/XA provides three methods of serializing resources: the ENQ-DEQ discipline, system provided locks, and GRS. The mechanisms are useful in both single CP and multiple CP, as well as multiple machine (complex) configurations.

ENQ-DE

QThe ENQ macro is performed on a name of a resource; this name is the means by which the operating system regonizes the resource. Users can ENQ exclusively, or indicate that they are willing to share. If the user ENQs exclusively, then that user must wait until all current ENQs on the resource have been DEQed. A shared ENQ capability enables users to allow others to access the resource. Both reads and writes are allowed for a shared ENQ. In both ENQ cases, the ENQ is released by the DEQ macro. Whereas the ENQ macro locks a resource, the RESERVE macro locks a DASD volume. The RESERVE lock is released by the DEQ macro. The lock is a bit on the controller for the volume, and only locks other systems in a complex out of that volume (i.e., other tasks in the system, on the same or other CPs, can still access the volume).

Locks

MVS/XA also provides locks, a mechanism to exclusively acquire system resources. There are two categories of locks: global locks, which are used on resources related to more than one address space, and local locks, which reserve resources local to a single address space. A global or local lock can be one of two types: spin or suspend.

Each lock type has its own lock manager composed of routines that control the use and behavior of the locks. The lock types are differentiated by the action that the processor attempting to lock the resource takes when the resource is already locked or otherwise unavailable.

A spin lock causes the requestor to poll the lock until the lock is released, at which time the requestor gains the lock on the resource. A suspend lock, on the other hand, causes the requesting unit of work to be suspended until the lock is released; this will free the processor that the requestor was running on for other work. The lock manager is responsible for waking up the suspended process when the lock becomes available.

When a lock mechanism exists on a system, it is possible for two processes to each hold locks on a resource the other process requests, resulting in a deadlock. To prevent this situation, a hierarchy of locks has been established so that a process can only request (and obtain) locks that are higher in the hierarchy than all locks it currently holds. Figure 4 lists all of the locks in high to low priority order.

Lock Name	Category	Type	Description
RSMGL	Global	Spin	Serializes RSM global resources.
VSMFIX	Global	Spin	Serializes VSM work areas for global fixed subpools.
ASM	Global	Spin	Serializes ASM resources on an address space level.
ASMGL	Global	Spin	Serializes ASM resource on a global level.
RSMST	Global	Spin	Serializes RSM control blocks on an address space level when it is not known which address space locks are current held.
RSMCM	Global	Spin	Serializes RSM common area resources (PTEs).
RSMXM	Global	Spin	Serializes RSM control blocks on an address space level wh serialization is needed to a second address space.
RSMAD	Global	Spin	Serializes RSM control blocks on an address space level.
RSM	Global	Spin	Serializes RSM functions and resources on a global level.
VSMPAG	Global	Spin	Serializes the VSM work area for global pageable subpools
DISP	Global	Spin	Serializes the ASCB dispatching queue.
IOSYNCH	Global	Spin	Serializes, using a table of lockwords, IOS resources.
IOSUCB	Global	Spin	Serializes access and updates to the UCBs. There is one IOSUCB lock per UCB.
SRM	Global	Spin	Serializes SRM control blocks and associated data.

TRACE	Global	Spin	Serializes the system trace buffer.
CP	Global	Spin	Provides system-recognized (legal) disablement. This lock has no hierarchywith respect to the other spin type locks, but once obtained, no suspend locks can be obtained.
CMSSMF	Global	Suspend	Serializes SMF functions and control blocks. This lock, QMSEQDQ and CMS are all equal to each other in the hierarchy.
CMSEQDQ	Global	Suspend	Serializes ENQ/DEQ functions and control blocks.
CMS	Global	Suspend	Serializes on more than one address space where this serialization is not provided by one or more of the other global locks.
CML	Local	Suspend	Serializes functions and storage within an address space other than the home address space. There is one CML lock per address space. This lock and LOCAL are equal to each other in the hierarchy.
LOCAL	Local	Suspend	Serializes functions and storage within a local address space There is one LOCAL lock per address space.

Figure 4

Global Resource Serialization

Global Resource Serialization (GRS) is a mechanism which allows many different machines in a complex (see page 8, "JES COMPLEXES") to serialize on a specific resource. Before the advent of GRS, the RESERVE instruction was used, which effectively locked the entire volume the resource was on. GRS allows serialization on specific resources on the volume. GRS uses the CTCs to communicate among the machines in the complex in reserving the resources. GRS includes many different mechanisms that enable one to detail the state of the system with respect to the resources affected by GRS. These mechanisms include ways to dump the GRS control structures, format the GRS data, scan the resource information directly, and display the resource contention information.

In order to use GRS, the user must specify a scope of "SYSTEMS" on the ENQ instruction. In order to provide compatibility with older programs, there are three exits that GRS invokes. The first exit is called the inclusion exit and is invoked when a scope of "SYSTEM" is specified on an

ENQ or DEQ. An associated list, called the SYSTEM inclusion list, contains names of resources that should be serialized via GRS. If the name of the resource is on this list and not on the SYSTEMS exclusion list (described next), GRS will be used to serialize the resource. Otherwise, local serialization is performed.

The second exit is the SYSTEMS exclusion exit, which uses an associated SYSTEMS exclusion list. If a resource is specified on this list and the "SYSTEMS" scope is used on the ENQ or the DEQ, the scope will be changed by GRS so that local serialization is performed.

The third exit is the RESERVE conversion exit, which also has an associated resource name list. If the resource name is specified on the SYSTEMS exclusion list, an ENQ with SYSTEM scope will be issued, and the RESERVE instruction will be issued. If not on the exclusion list and on the RESERVE resource name list, the resource will be ENQed with a SYSTEMS scope, and the RESERVE will be suppressed by GRS. If the resource is neither on the exclusion list nor the resource name list, that resource will still be ENQed with a SYSTEMS scope, but the RESERVE instruction will be issued.

Exit Routines

An exit is a defined point in a system program where that program calls another program. IBM supplies a program which performs default processing when an exit is reached in a system program. The IBM supplied program is designed to be replaced by an installation-written routine or to be disabled. For example, an exit is supplied in JES to allow the installation to supply their own algorithm for searching the job queue for jobs to be processed. Installation-written exit routines are not allowed in the evaluated configuration.

SVCs in MVS/XA with RACF

There are 139 different SVCs in MVS/XA with RACF, which are divided into five types: 1, 2, 3, 4, and 6. All SVCs run in the supervisor state, and hold protection key 0. The use of some SVCs is also APF authorized (see page 54, "Authorized Programs"); these are summarized below. An SVC can hold and acquire locks (see page 47, "Locks"); the majority of these routines are entered with locks specified in the SVC table. An SVC can call another SVC, if the caller does not hold a lock.

The code for types 1, 2, and 6 resides in the nucleus, while the code for types 3 and 4 resides in the Link Pack Area (see page 28, "Address Spaces"). Type 1 SVC routines are always entered with the LOCAL lock, even if it is not specified in the SVC table. Type 3 SVCs consist of one load module, while type 4 SVCs consist of more than one load module. Type 3 and 4 SVCs must fix their pages in real storage to avoid disabled page faults; i.e., page faults occurring after the SVC has acquired a disabled lock (any lock other than LOCAL, CMS, or CMSEQDQ). A type 6 SVC must always run with interrupts disabled, and must not enable them during its execution. Type 6 SVCs also may not be suspended for a lock request.

The SVC table resides in the system area, and contains eight bytes of information regarding each SVC. This information includes an entry point for the routine, the type and function code (authorized or not), locks to be acquired before entering the routine, and the addressing mode indication (24-bit or 32-bit).

The following is a list and short description of all of the SVCs that are restricted to invokers who are in supervisor state, keys 0-7, or are running APF authorized.

SVC 15: ERREXCP macro

> This is a type 1 SVC, which holds the LOCAL, IOSUCB, and IOSCAT locks. Its function is to post an error status to the ECB on a return from IOS.

SVC 32: (no macro name)

> This is a type 4 SVC, which acquires a LOCAL lock. Its function is to do initial allocation of space on DASD.

SVC 39: LABEL macro

> This is a type 3 SVC which holds no locks. Its function is to write the label to a tape.

SVC 52: RESTART macro

> This is a type 4 SVC which holds LOCAL, CMS, and SALLOC locks. It function is to restart the system from a checkpoint data set under the Checkpoint/Restart facility of MVS.

SVC 59: OLTEP macro

> This is a type 3 SVC which holds LOCAL and CMS locks. Its function is described on page117, "DIAGNOSTICS".

SVC 76: (no macro name)

This is a type 3 SVC which holds no locks. Its function is to format and write out various hardware and selected software errors to SYS1.LOGREC.

SVC 83: SMFWTM or SMFEWTM macro

This is a type 3 SVC and holds no locks. Its function is to complete and write the information contained in all SMF records out to the SMF data set.

SVC 85: DDRSWAP macro

This is a type 3 SVC which holds the LOCAL lock. Its function is to interface with a user-written routine that manages NSL tapes; the SVC verifies these tapes, and determines if it needs to be re-positioned.

SVC 86: ATLAS macro

This is a type 4 SVC which holds no locks. ATLAS is a disk recovery system, in which the module represented by this SVC is the first load module. The SVC's primary responsibilities involve determining if the error is correctable, and also acquiring information about the disk volume the error is on (such as the number of available alternate tracks).

SVC 104: TOPCTL macro

This is a type 4 SVC which holds no locks. It is used exclusively as an interface to TCAM. TCAM is not a part of the evaluated configuration, so this SVC is never called.

SVC 107: MODESET macro

This is a type 6 SVC, which holds no locks. Its function is to change the mode of the system by altering the information in the old PSW (i.e., the one that the program which invoked the SVC was using). The information that can be affected is the state (supervisor or problem), the PSW protection key, or the key mask.

SVC 123: PURGEDQ macro

This is a type 2 SVC which holds the DISP lock. Its function is to purge Service Request Blocks (SRBs; see page41, "Dispatcher") off of the service manager queues, ensuring that suspended SRBs have completed their processing.

SVC 126: MSS Interface

This is a type 3 SVC which holds the LOCAL and CMS locks. The function of the SVC is to take the argument block and issue the TESTAUTH macro to verify the user is authorized (see page 54, "Authorized Programs").

SVC 130: RACHECK macro

This is a type 3 SVC which holds no lock. Its function is described on page 76, "RACF Interface to MVS".

SVC 131: RACINIT macro

This is a type 3 SVC which holds no lock. Its function is described on page 76, "RACF Interface to MVS".

SVC 132: RACLIST macro

This is a type 3 SVC which holds no lock. Its function is described on page 76, "RACF Interface to MVS".

SVC 133: RACDEF macro

This is a type 3 SVC which holds no lock. Its function is described on page 76, "RACF Interface to MVS".

Authorized Programs

Although RACF contains privileges which can be used by certain subjects, MVS/XA recognizes a class of programs which are "authorized". A program is authorized if one of following conditions is true: the program is in supervisor state, the program runs with a system key (keys 0-7), or the program is APF authorized.

The Authorized Program Facility (APF) enables certain programs to have special privileges. In order to gain these privileges, an APF authorized program must fulfill two conditions: it must be loaded from an authorized library, and it must be linked with AC=1 (authorization code).

Authorized libraries are specified in a configuration data set at Initial Program Load. They include SYS1.LINKLIB, SYS1.SVCLIB, and SYS1.LPALIB. Anyone can link a program with AC=1; however, the program is not authorized until it is put into one of the authorized libraries. When an authorized program calls an unauthorized program, the authorization is revoked so that even if the unauthorized programs returns to its caller (the authorized program), the (formerly) authorized program will not have its special privileges (i.e., will act just like an unauthorized program). In addition, when an authorized program is dispatched for a TSO/E (foreground and background) address space, all other tasks in that address space are marked non-dispatchable until the authorized program is finished.

JOB ENTRY SUBSYSTEM 2

The job entry subsystem (JES) is used to manage jobs before and after execution. The function of JES is to screen jobs before admission to the system and to handle their termination. This section defines some terms relating to JES2 (one version of JES) and describes the functions of JES2.

Job Control Language, Jobs, and Tasks

The Job Control Language (JCL) supplied by IBM are statements which define the work to be done for a job. A job is a unit of work given to an initiator, and all the job steps from that job will execute in that initiator (see page 36, "Address Space Creation"). A job step is the smallest executable portion of a job, and it is made up of one or more JCL statements required to execute a program. Job steps execute sequentially.

Three important JCL statements are JOB, DD, and EXEC. The JOB statement marks the beginning of a job, tells the system how to process the job, assigns a name to the job, and contains information like password, userid, and groupid (see example below). The DD statement is used to identify a data set and to specify the input and output resources needed for the data set. Information placed on the DD statement includes the number of copies, the access method, the destination, the name, and the volume serial number of the data set. The EXEC statement marks the beginning of each job step in the job. This statement is used to identify the program, cataloged procedure, or in-stream procedure that is to be executed. Some items found on the EXEC statement include the program name, the length of time the step may execute, and the type of storage required for the step. Other JCL statements are used for marking the end of a job and specifying options for printing data sets. An example of a JCL program follows:

```
//MYJOB  JOB          ,’JOHN SMITH’,TIME=(4,30),
//       USER=JSMITH,PASSWORD=ASDF
//STEP1  EXEC         PGM=CREDIT
//CHARGE              DD  DSN=CHARGE.FILE,DISP=SHR
//STATEMNT DD         SYSOUT=A, DCB=(LRECL=132,
//       BLKSIZE=1320,RECFM=FB)
//**************************************************
//STEP2  EXEC         PGM=DEBIT
//MASTER              DD  DSN=MASTER.FILE,DISP=SHR
//PRINT  DD           SYSOUT=*
//
```

In the above example there are two job steps in the job MYJOB, STEP1 and STEP2 with their corresponding DD statements. The job was submitted by userid JSMITH with a password of ASDF. The first step, STEP1, executes the program named CREDIT and uses one data set as input and a SYSOUT data set for output. The second step, STEP2, executes program DEBIT, uses one data set as input, and uses SYSOUT as its output data set.

A task is a dispatchable unit of work in a job. MVS/XA breaks a job up into tasks and each task is processed as efficiently as possible. All tasks compete with one another for system resources. The progress of tasks is controlled through the system by allocating resources (other than I/O devices) and maintaining current information about each task. A task can also be defined as a request for the execution of some code. A new task is created for every job step in a job, and more than one task may execute in an address space at the same time.

Jobs of similar characteristics and processing requirements can be assigned to a job class. There are 36 job classes, A-Z and 0-9 (the names have no inherent meaning). A job class is assigned to one or more initiators, and a job can only be started on an initiator if the job's class is assigned to the initiator. The job class is specified on the JOB statement or, if not specified, assigned based on the device through which the job entered JES2. If an initiator is assigned job class H and job class H has been defined by the installation as jobs with high I/O requirements, then a user can specify job class H on the JOB statement for an I/O intensive job.

Spool Data Sets

JES2 uses spool data sets to hold a job and its associated information between various JES2 stages. The spool data set is a physical sequential data set residing on a direct access device such as a disk. A spool data set is contained within a spool volume, and the spool volume is defined to JES2. There are different types of spool data sets, for example SYSIN, SYSOUT, and dump. A SYSIN data set contains data entered into the system input stream with JCL statements. During the output phase the jobs are queued to the SYSOUT data set for that job on the spool device. SYSIN and SYSOUT must be either BSAM or QSAM data sets (see page 65, "Access Methods"). A dump data set is a backup of the spool data set. Because spool dumps contain clear text versions of all JOB statements, they are protected with RACF (see page 96, "DISCRETIONARY ACCESS CONTROL"). In the case of a complex the different copies of JES2 use the same spool data sets.

Job Processing

JES2 in MVS/XA with RACF performs the following basic functions: reading jobs into the system, converting jobs to internal form, selecting jobs for execution, preparing jobs' output for printing, placing jobs' output on the hardcopy queue, and purging jobs from the system. These six functions correspond to the six phases in JES2 processing. More detailed descriptions of these phases follow.

In a JES complex the job queues are shared. This allows a job's phases to be performed on any machine in the complex. For example, one machine may convert a job's JCL and place the job on the spool while another machine may retrieve the job from the spool and execute it. Each JES2 phase will complete on only one machine.

Input

Input streams can come from a remote terminal, a magnetic tape or a direct access device. JES can also receive input streams from internal readers which are data sets that other programs can use to submit jobs, control statements, and commands. Any job can use an internal reader to pass input streams to JES. There can be up to 255 internal readers on a system, and the number of internal readers affects the number of jobs JES2 can receive simultaneously. A job ID is assigned to each input stream as it comes in. The job's JCL, optional JES2 control statements, and input data are placed into spool data sets.

Conversion

JES2 uses a converter program running in JES's address space to convert a job's JCL into internal text so that both JES2 and MVS can recognize it. The translated text is then stored on the job queue or on the spool data set. If the converter finds errors the job is placed on the output queue, otherwise it is queued for execution.

Execution

When a free initiator requests a job, JES2 selects a job from the job queue. A JES initiator is a logical initiator defined at JES initialization corresponding to a system initiator. The job queue resides in the JES2 address space in main memory and a copy is kept in the checkpoint data set. In a complex the job queue resides on the spool volume. The job queue contains the following:

- jobs waiting to run
- jobs currently running
- jobs waiting for their output to be produced
- jobs having their output produced
- jobs waiting to be purged from the system

To process the jobs on the job queue, JES2 waits for an initiator to ask for a job. JES2 keeps track of what job class or job classes are assigned to the initiator and in what order the job classes should be searched for a job. Jobs are selected by priority and then passed to the initiator. Once an initiator receives a job, the job is placed in the initiator's address space and the job steps begin executing.

Output

JES2 controls all SYSOUT processing. JES2 gathers the output data by output class, process mode, and setup characteristics and places the job on the job output table.

Hardcopy

Output is selected from the job output table by priority. The use of priority for output is an installations choice with a default of none. If used, a user can specify the priority level pwd

(1-255) at which the SYSOUT data set enters the output queue. If the priority is not specified on the JOB statement, an output priority is calculated based on the number of lines of print and the number of pages. The job's print data sets are queued to the SYSOUT spool data set.

Purge

After all processing for a job is finished JES2 releases all spool space belonging to the job and notifies the operator that the job has been purged. The operator can use the Message Processing Facility to suppress these messages at the console. The messages are then stored in a log data set instead of being sent to the console.

DATA FACILITY PRODUCT

The MVS/XA Data Facility Product (DFP) provides data management support, device support, program library management support, and utility functions for IBM processors which implement the System 370-XA architecture. When installed with the corequisite products, either MVS/System Product JES2 or JES3, the combination is an operating system environment called MVS/XA. DFP provides these services in support of MVS/XA objects. Definitions of these objects follow.

Device Volumes and Labels

Device volumes are the physical containers of data. Physical disk packs may contain one direct access volume while tape reels contain one (logical) tape volume per reel.

Direct Access Disk Volumes

Direct access disk volumes are used to store executable programs, including the operating system itself. Direct access storage is also used for data and for temporary working storage. A volume table of contents (VTOC) is used to account for each data set and available space on the volume.

Each direct access volume is identified by a volume label that is stored at track 0 of cylinder 0. Additional labels may follow the standard volume label. Each direct access volume is initialized by a utility program before being used on the system. The initialization program generates the volume label and builds the table of contents.

Although direct access devices differ in appearance, capacity, and speed, they are similar in data recording, data format, and programming. The recording surface of each volume is divided into many concentric tracks. Information is recorded on all direct access volumes in a standard format. Besides device data, each track contains a track descriptor record and data records. The track descriptor record contains: the location of the record by cylinder, head, and record numbers; its key length (for keyed or indexed data sets, or 0 if keys are not used); and its data length. A track overflow option allows a block of data that does not fit on the track to be partially written on that track and continued on the next (adjacent) track. This adjacent track must be allocated to the same data set, otherwise the write fails.

All data sets (a complete description of data sets follows) are allocated in increments of one or more tracks although the user may optionally specify cylinders or blocks as units of allocation. Space is allocated based on two quantities: primary and secondary. The primary quantity is the first extent (block) allocated. For partitioned data sets, it also includes the space necessary to hold the data set directory. The secondary quantity specifies the number of additional tracks, cylinders, or blocks to be allocated, if more space is needed. A direct data set is composed of only the primary space. All other direct access device data sets may be comprised of one primary extent and up to 15 secondary extents.

Magnetic Tape Volumes

Magnetic tape volumes are used to store data mainly for archival purposes. Labels are optionally used to identify these volumes and the data sets they contain. The system can process tapes with IBM standard labels, ISO/ANSI (International Standards Organization, and American National Standard Institute) labels, nonstandard labels, and no labels.

IBM standard tape labels consist of volume labels and groups of data set labels. The volume label is the first record on the tape. It identifies the volume and its owner. The data set groups precede (header labels) and follow (trailer labels) each data set on the volume, and identify and describe the data set. IBM standard labeled tapes contain a tapemark between the data set and its header and trailer labels, and a double tapemark after the last trailer label.

ISO/ANSI labels are similar to the formats of IBM standard labels. However, whereas ISO labeled tapes are coded in the International Standard Code for Information Interchange (SCII) and ANSI labeled tapes are coded in the equivalent American National Standard Code for Information Interchange (ASCII), IBM labeled tapes are coded either in the extended binary-coded-decimal interchange code (EBCDIC) or in binary coded decimal (BCD). ISO/ANSI labeled tapes contain a tapemark between the data set and its header and trailer labels, and a double tapemark after the last trailer label.

Nonstandard tape labels can have any format but users must provide a routine for their processing. Nonstandard labeled tapes may contain single tapemarks between data sets. Unlabeled tapes contain only data sets separated by tapemarks and end with double tapemarks.

Data sets are placed on magnetic tape immediately following the header label (of a labeled tape) and are written sequentially from the beginning of tape. DFP maintains the integrity of the tape volume label (the header label) while allowing the authorized user(s) full access to the remainder of the volume. When the hardware senses the beginning of tape, the system restricts modifications of the tape label to authorized programs.

Volume Table of Contents

Each data set stored on a volume has its name, location, organization, and control information stored in the tape volume table of contents (TVTOC) for magnetic tape volumes, and in the volume table of contents (VTOC) for direct access disk volumes. In addition, each data set entry in both tables contains a discrete profile flag, creation date, and the last reference date. The TVTOCs are stored in the RACF database while the VTOCs are stored on the disk volumes they represent.

The direct access device storage management routines control allocation of space on direct access volumes through the VTOC of that volume, and through the VTOC index if one exists. The VTOC resides in a single extent (as a contiguous sequential data set) anywhere on the volume after cylinder 0, track 0. Its address is located in the standard label of that volume. The VTOC is composed of data set control blocks (DSCBs) that correspond either to a data set currently residing on the volume, or to contiguous, unassigned tracks on the volume. DSCBs for data sets describe their characteristics. DSCBs for contiguous, unassigned tracks indicate their starting location and length.

The VTOC index is a VSAM data set residing on the same volume as the VTOC. It contains an index of data set names in the VTOC and free space information. Its name has the form of SYS1.IXVTOC.Vnnnnnn where nnnnnn represent the volume name.

Catalogs

In order to facilitate data set storage and retrieval, MVS/XA through a specialized DFP service provides for cataloging of data sets using the integrated catalog facility. The catalog structure consists of a master catalog and user catalogs. There is one master catalog on each system. This catalog is a virtual storage access method (VSAM) key-sequenced data set containing volume ownership and security information, data set ownership and security information, and other information for VSAM and non-VSAM data sets. (VSAM and non-VSAM data sets are described below.)

The master catalog contains pointers to user catalogs. These, in turn, contain pointers to the respective VSAM volume data sets or to VTOCs residing on the target volumes which ultimately point to the VSAM and non-VSAM data sets, respectively.

DFP provides several functions for manipulating catalogs. These include creating a catalog, converting a catalog, defining objects in a catalog, modifying a catalog, deleting catalog entries, and copying, merging, splitting, backing up and listing a catalog. The catalogs are only indirectly accessible to unprivileged users.

Data Sets

A data set (file) is a collection of logically related data records that are stored on a volume. Data sets can reside on direct access (disk) volumes or on tape volumes. DFP supplies the system and the users with a wide range of data set and volume manipulation functions.

All data sets cataloged within one catalog (see page 61, "Catalogs") must have unique names. This restriction holds for all the data sets recognized only by name (specified without the volume identification) and all the data sets residing on a given volume. (When referring to a cataloged data set, a user needs only to specify the data set name which must be unique to the catalog. When referring to an uncataloged data set, a user must specify the volume holding the data set. That data set must be unique to that volume.) A data set name is one or more simple names joined together with periods. The first name is called the high level qualifier and it is often identical to the userid of a user owning the data set. Each simple name consists of from 1 to 8 characters, the first of which must be alphabetic. The length of the data set name cannot exceed 44 characters.

In summary, the following is the MVS/XA storage hierarchy. Bytes of data are grouped to form records of either fixed or variable length. Records make up blocks which make up tracks. A data set occupies at least one track. Data sets are stored on volumes and may also span volumes. A data set location, organization, protection, and some VSAM data set names are found in a VTOC, while most of data set names, ownership, additional protection information, and volume location, reside in a catalog. The master catalog may contain information about all other catalogs and volumes.

Data Set Types

MVS/XA recognizes four types of data sets: temporary data sets, spool data sets, VSAM data sets, and non-VSAM data sets. Some temporary data sets are implemented using Virtual I/O (VIO) and appear as extensions to user address spaces. Others simply are VSAM or non-VSAM data sets kept under strict control of the system or a subsystem. They are system-owned, not shared, and never cataloged. They are allocated to jobs only for the duration of a job's existence. For a complete discussion of VIO and spool data sets, refer to page 35, "Virtual Input/Output" and page 56, "Spool Data Sets". VSAM data sets are used by the system primarily for paging and swapping data sets and in implementing volume catalogs. They are also used by user applications where the user supplies most of the access method details. Non-VSAM data sets make up a large majority of data sets used by users to store and share data.

VSAM Data Set Organization

VSAM data sets can be organized in one of three ways, each with a different set of characteristics. In a key-sequenced data set, records are loaded in key sequence. Each record must have a key, and the ordering of the records is determined by the numeric value of the keys. New records are added in key sequence. In an entry-sequenced data set, records are loaded in sequential order as they are entered. New records are added at the end of the data set. In a relative record data set, records are loaded according to a relative record number that can be assigned either by VSAM or by the user program. VSAM-numbered records are added at the end of the data set; user-numbered records can be added in relative record number sequence.

The format of a VSAM data set record is different from that of other data sets. All VSAM data set records are stored in control intervals. A control interval is a continuous segment of auxiliary storage. With key-sequenced data sets, the user can gain access to a record by specifying its key or its relative byte address. With entry-sequenced data sets, the user can gain access to a record only in the order the records were added to the data set. Finally, with relative record data set, the user can gain access to a record only by specifying its relative record location.

Non-VSAM Data Set Organization

Direct access devices may contain four types of non-VSAM data sets: direct data sets, sequential data sets, indexed sequential data sets, and partitioned data sets.

Direct Data Sets

There are two types of addresses that can be used to store and retrieve data in a direct data set: actual addresses and relative addresses. The actual address of a record contains a 1-byte binary number specifying the relative location of an entry in a data extent block (DEB). The DEB is created by the system when the data set is opened. Each extent entry describes a set of consecutive tracks allocated for the data set. The actual address further contains three 2-byte binary numbers specifying the cell, cylinder, and head number for the record (its track address). A 1-byte binary number specifying the relative block number on the track is also a part of the actual address.

There are two kinds of relative addresses each of which may use an actual key: relative block addresses and relative track addresses. The relative block address is a 3-byte binary number that describes the position of the block relative to the first block of the data set. This data set can be allocated with noncontiguous sets of blocks without affecting the relative block address.

The relative track address contains a 2-byte binary number specifying the position of the track relative to the first track allocated for the data set. The track position for the first track is 0. Allocation of noncontiguous sets of tracks does not affect the relative track address. This address also contains a 1-byte binary number specifying the number of the block relative to the first block on the track previously specified. This number for the first block of data on a track is 1.

In addition to the relative track or the relative block address, the address of the virtual storage location containing the record key may be specified. The system then computes the actual track address and searches for the record with the correct key.

Sequential Data Sets

Sequential data sets may reside on both direct access devices and tape devices. They have the same limited set of characteristics: records are written sequentially from the beginning of the data sets; new records are added to the end of the data set (extending the data set); records may be updated but they cannot be deleted or their length modified. Information must be searched from the beginning of the data set.

Indexed Sequential Data Sets

Indexed sequential data sets offer many advantages over the sequential data sets. The data set can be read or written sequentially, individual records can be processed in any order, records can be deleted, and new records can be added.

The records in an indexed sequential data set are arranged according to collating sequence by a key field in each record. Each block of records is preceded by a key field that corresponds to the key of the last record in the block.

Indexed sequential data sets reside on direct access storage devices. They may occupy three different areas. The prime area contains data records and related track indexes. The overflow area contains records that overflow from the prime area when new data records are added. The index area contains master and cylinder indexes associated with the data set. It exists for a data set that has a prime area occupying more than one cylinder.

Partitioned Data Sets

Partitioned data sets can only be stored on direct access devices. A partitioned data set is divided into sequentially organized members, each composed of one or more records. Each member has a unique name stored in a directory that is a part of the data set. The records of a given member are written or retrieved sequentially. The individual members can be added or deleted as required. However, deleted space is not reused until the entire partitioned data set is copied or compressed.

The directory, located at the beginning of the data set, is made up of 256-byte records containing an entry for each member. Each directory entry contains the member name and the starting location of the member within the data set. The directory entries are arranged by name in alphameric collating sequence. The starting location of each member is recorded by the system as a relative track address (from the beginning of the data set). If there is not sufficient space available in the directory for an additional entry, or not enough space available within the data set for an additional member, or no room on the volume for additional extents. no new members can be stored.

Non-VSAM Data Set Access Techniques

There are two techniques a program can use to access the records in a non-VSAM data set: the queued access technique or the basic access technique. The queued access technique is used when the sequence in which records are to be read or written is known to the access method. The system can anticipate which records are needed and make them available through buffering. The access method does not return control to the program utilizing this technique until the requested I/O operation has been completed.

The basic access technique is used when no assumption can be made about the sequence in which records are to be processed. The basic technique allows access to any records in the data set. No grouping of records takes place and no anticipation of future I/O requests occurs. The program utilizing this access technique must test for the completion of the I/O operation because the access method returns control to the program before the I/O operation is completed.

Access Methods

Corresponding to the data set organizations, there are four types of access methods. An access method is a system service which a user may invoke to access data stored in a data set. As mentioned earlier, VIO is utilized when accessing temporary data sets. JES uses its own routines in handling the spooled data sets. The remaining two types of data sets -- VSAM and non-VSAM -- are handled with the virtual storage access method and the conventional access methods, respectively.

VSAM

The virtual storage access method is an access method used to organize system and user data and to maintain information about that data in a catalog. VSAM performs catalog management and record management.

VSAM is specifically designed to take advantage of virtual storage. VSAM is used to access disk data; it runs in virtual storage and uses virtual storage to buffer I/O operations. VSAM does not use the EXCP processor. VSAM employs its versions of queued and basic access techniques allowing it to process the three types of data sets previously described.

Information is requested from or supplied to VSAM data management in logical records. As mentioned earlier, logical records of VSAM data sets are stored differently from logical records in non-VSAM data sets. VSAM uses control intervals to contain records. Whenever a record is retrieved from direct access storage, the entire control interval containing the record is read into a VSAM I/O buffer in virtual storage. From the VSAM buffer, the desired record is transferred to a user-defined buffer or work area in that user's address space.

A control interval is a continuous area of direct access storage that VSAM uses to store data records and control information that describes the records. The control intervals in a VSAM data set are grouped together into contiguous areas of direct access storage called control areas. A VSAM data set is actually composed of one or more control areas. The maximum size of a control area can vary between one track and one cylinder of DASD storage.

Conventional Access Methods

Conventional access methods move data to and from non-VSAM data sets. These access methods are identified by the technique they employ and the type of data organization to which they apply. MVS/XA supports the six types of conventional access methods.

Data Set	Access Methods	
Organization	Basic Technique	Queued Technique
Sequential	BSAM	QSAM
Partitioned	BPAM	
Indexed Sequential	BISAM	QISAM
Direct	BDAM	

Other DFP Functions

As mentioned in the introduction to this section, DFP also provides program management and system support functions. The linkage editor combines previously compiled or assembled object modules into a program ready to be loaded and executed. It also allows users to edit program modules and selectively replace sections within the program. Program fetch is a mechanism for reading a load module into virtual storage and relocating any address constants in the module. The loader combines the basic editing functions of the linkage editor and the loading functions of program fetch into one step. It loads for execution object modules produced by language translators, and load modules produced by the linkage editor.

The checkpoint/restart facilities gather and record information about the status of a job and its related control blocks to allow a restart, should one become necessary. Execution resumes at the beginning of a job step (step restart) or from a place within a job step (checkpoint restart). Operators control step restarts. User application programs are responsible for taking checkpoints for possible future restarts.

DFP also provides a set of general purpose data set utility programs for copying, merging, loading, unloading, reblocking, comparing, updating, and printing data sets. System utility programs are also supplied. They provide the means to label magnetic tapes, locate and assign alternate tracks on a disk, rebuild defective records, list partitioned data set directories, and modify system control data.

ACF/VIRTUAL TELECOMMUNICATIONS ACCESS METHOD

Advanced Communications Function for the Virtual Telecommunications Access Method (ACF/VTAM) provides the data communication capability used by MVS/XA to connect terminals to the system. ACF/VTAM (also referred to as VTAM) allows a VTAM application program (i.e., Terminal Communication Address Space) to communicate with terminals using symbolic names. The use of symbolic names by an application program reduces the complexity of the application program. These symbolic names allow a VTAM application program to be unaware of how a terminal is attached to the communication network (directly or through a controller), where the terminal is located (network address), or if any intermediate devices exist. VTAM does not enforce an access control policy on communication from a VTAM application program to a terminal. Therefore, both VTAM and the VTAM application program must be trusted.

The components within a data communication network include each system (i.e., 3090, 4381), cluster controllers (i.g., 3174) and terminals (i.g., 3178 and 3179). Components must be defined to VTAM by a system administrator before the components may become active. Defining a component causes a symbolic name, a local address, the identity of an initial application program (optional), the local address of the controlling physical unit, and physical characteristics (i.e., screen size, line length) to be stored in the SYS1.VTAMLST data set. The symbolic name specified for a terminal must match its RACF terminalid. The initial application program is the VTAM application program to which a terminal is automatically connected upon power on.

SYS1.VTAMLST is a partitioned data set, with each member identifying a major node, a VTAM application program or another system within a complex. A major node is a set of resources that can be independently activated or deactivated as a group. A major node is a channel-attached 3274 cluster controller. Changes to SYS1.VTAMLST become effective each time an inactive major node is activated.

Network Addressable Units

The VTAM data communication network includes components which can transmit or receive data. These components are known as network addressable units (NAUs). An NAU can be either a device, a program, or a portion of ACF/VTAM. There are three types of network addressable units: system service control points (SSCPs), physical units (PUs) and logical units (LUs).

An SSCP is a portion of VTAM which runs within the host processor and manages the network. The SSCP takes part in network start up and shutdown, initiating and terminating communication between NAUs, and network recovery.

A PU is the programming within a physical device which controls the device, and sometimes other devices. Each physical device is controlled by a PU although this PU need not be unique for each physical device. Every system and cluster controller contains a PU. The controlling PU of a terminal may either be within the terminal or within the controlling unit (i.e., cluster controller, or processor) to which the terminal is attached. A PU is known to VTAM by its channel unit address.

An LU is programming or built-in logic associated with either a terminal or an application program. The VTAM considers an LU as the source of a request coming into the network (i.e, from either an application program or a terminal). Figure 5 depicts a sample VTAM data communication network and identifies its NAUs.

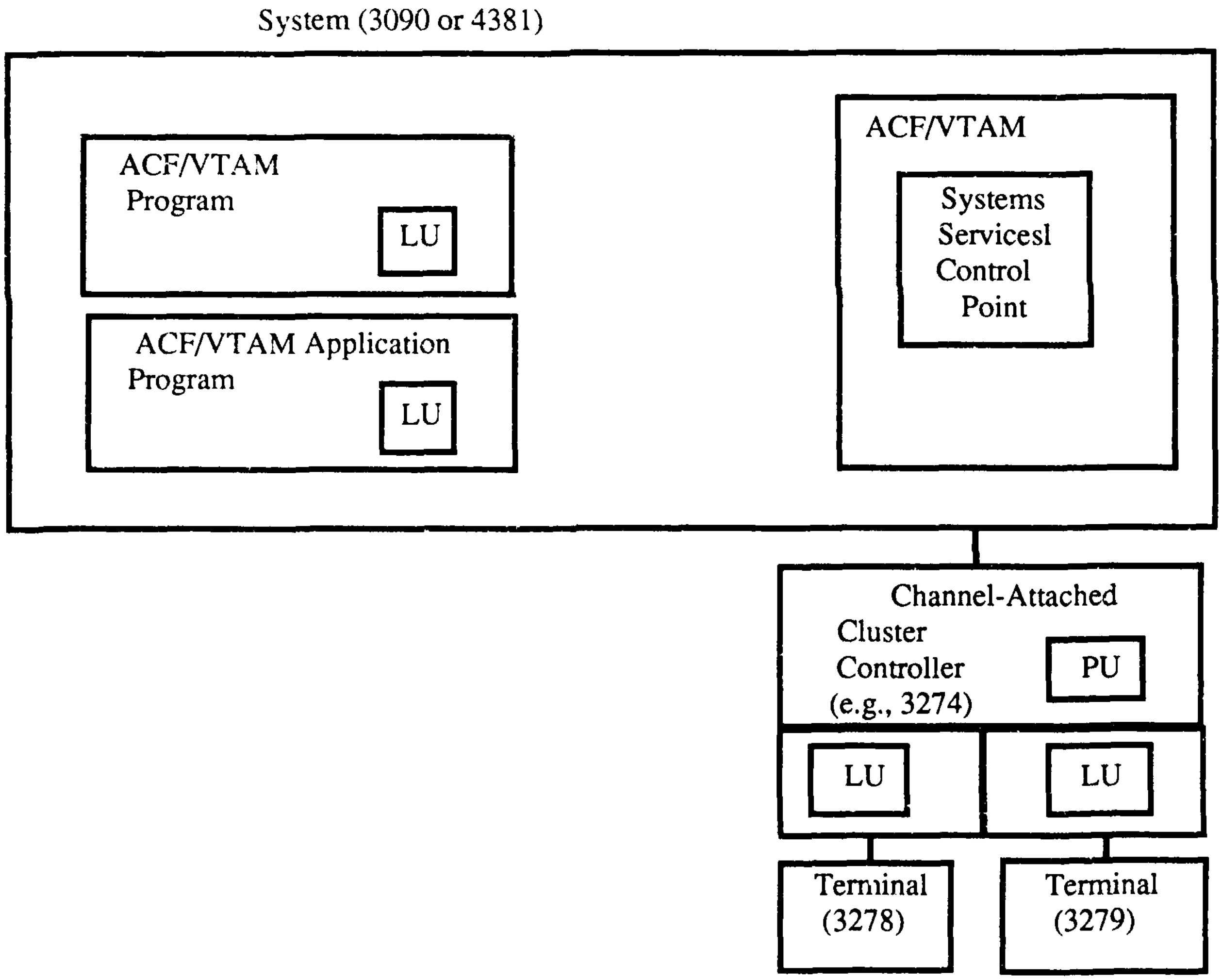


Figure 5.

Sessions

Before communication between two NAUs occurs, a "session" must be established. A session is the logical connection established between two NAUs. There are four types of sessions that can be established: SSCP-PU sessions, SSCP-LU sessions, LU-LU sessions, and SSCP-SSCP sessions.

An SSCP-PU session is established for each PU when the network is initiated. These sessions remain active until a PU is no longer in service (e.g., powered down). After a SSCP-PU is established, SSCP-LU sessions are established for each LU controlled by the PU.

An LU-LU session is established whenever one LU wishes to begin communications with another LU. An LU associated with a terminal may have only one LU-LU session, while an LU associated with a VTAM application program may have many LU-LU sessions.

Finally, an SSCP-SSCP session occurs only between two systems within a complex. Each processor contains one SSCP, which controls a set of PUs and LUs. The set of PUs and LUs controlled by an SSCP is known as a domain. A SSCP-SSCP session is used to communicate across domains. SSCP-SSCP sessions allow an LU to transparently communicate with an LU in another domain. Communication between domains is performed using a cross domain link.

VTAM Application Program Interface

ACF/VTAM consists of a VTAM address space and a VTAM application program interface. The VTAM address space is responsible for maintaining information describing the VTAM configuration (e.g., physical address of a terminal). The VTAM address space is also the address space that is used to the create channel control programs that pass information to and from the terminals. The VTAM application program interface (API) is the part of VTAM that executes in each address space. The VTAM API gets scheduled as a task in each address space and copies information from the address space's common area to an address space's private area.

The ACF/VTAM API provides four types of macros that allow application programs to request services. These four types of macros are declarative macros, manipulative macros, ACB-based macros, and RPL-based macros.

Declarative macros are used by an application program to buildcontrol blocks. These control blocks describe the application program (Access method Control Block -- ACB), its exits (EXit LiST -- EXLST), its session communication requirements (Node Initialization Block -- NIB), and its required parameters (Request Parameter List -- RPL).

Manipulative macros access or modify fields within control blocks. These macros provide a more consistent and convenient method for manipulating control blocks than application issued assembler instructions.

There are only two ACB-based macro instructions: OPEN and CLOSE. These macros are used to inform VTAM that the application program is beginning or ending its use of VTAM services.

Finally, RPL-based macros are used to request session establishment, data transfer and program operator control. Each of these macros accepts an RPL control block which contains parameters for the macro. RPL-Based macros can be further categorized by function as follows:

- Session Establishment as Primary LU
- Session Establishment as Secondary LU
- Communication
- Assist in Session Establishment and Communication
- Program Operator Control

The VTAM API provides macros that allow an application program to send information via VTAM. The VTAM API copies data and macro parameters from the private area of an address space into CSA. Once in CSA the data and parameters are visible to the VTAM address space. VTAM collects multiple requests for a controller before generating the channel control words (see page 20, "The Channel Subsystem"), which are passed to IOS to perform the terminal I/O.

Virtual Telecommunications I/O Coordinator

The TSO/E interface to VTAM is more complex than the interface just described. TSO/E is a program product that is older than VTAM, and the interface between TSO/E and VTAM is not directly compatible. Therefore, an interface program was developed called Virtual Telecommunications I/O Coordinator (VTIOC). VTIOC is responsible for interpreting the information flowing between TSO/E and VTAM, and translating it into the form accepted by each program product.

Application Programs

As mentioned earlier, when a terminal is powered on, a session is established between the terminal and the application program specified when the terminal was defined to VTAM. Commonly this application program is the Terminal Communication Address Space (TCAS), which handles logins for TSO/E (see page 72, "Terminal Control Address Space"). If no application program is specified for a terminal, the user instructs VTAM to connect the terminal to a VTAM application program. A VTAM application program is any program identified in the SYS1.VTAMLST data set. The program identifiers in the SYS1.VTAMLST data set instruct VTAM to connect the terminal to the specified application program. Every terminal that can be connected to TSO/E must have a unique application program identity (e.g., TCAS00001, TCAS00002). This identity is derived from the application program specified in the SYS1.VTAMLST data set.

When a terminal is powered on VTAM establishes a session between an SSCP and the terminal's LU. If an application program is specified for the terminal VTAM establishes an LU-LU session between the terminal and the application program. In the evaluated system the only allowable VTAM application program is TCAS.

Terminal Control Address Space

TCAS is responsible for preparing an address space before the logon process completes. TCAS, as its name implies, executes within its own address space which is established during system IPL. If TCAS is specified as the terminal's default application program, VTAM is used to establish a session between the terminal and TCAS whenever the terminal is powered on or after a user logs off the terminal. TCAS then monitors inputs from the terminal, waiting for a LOGON request. During a LOGON request, TCAS processes the command and acquires an address space for the user (see page 36, "Address Space Creation").

Once the address space has been created, TCAS terminates its session with the terminal, and initiates a session between the terminal and the new address space. This new address space is executing a copy of TCAS and is referenced as TCASnnnnn (e.g., TCAS00001 or TCAS00002). TCAS continues monitoring the remaining powered on terminals for LOGON requests.

TIME SHARING OPTION/EXTENSIONS

Time Sharing Option Extensions (TSO/E) is IBM's terminal interface application available for the MVS/XA operating system. It supplies the required services to process user commands and batch jobs, while supplying this service in an interactive format for a time sharing system

TSO/E consists of several components which execute commands and interfaces with the system and user. These components are the Terminal Monitor Program (TMP), command processors, and the User Attributes Data Set. This TSO/E section also discusses the TSO/E interface to VTAM.

Terminal Monitor Program

In a TSO session, the Terminal Monitor Program (TMP) executes as a task under STC. TMP is responsible for obtaining TSO/E commands from VTAM, attaching command processors, and monitoring the execution of the command.

Specifically, TMP is attached under the LOGON/LOGOFF scheduler (part of STC) when it is called from the EXEC (execute) statement in the logon procedure. TMP executes as a task until an operator issues a STOP command or the user requests a LOGOFF. At that point, TMP is terminated and returns control to the LOGON/LOGOFF scheduler which terminates the session between VTIOC before returning control to STC.

During execution, TMP uses TSO service routines to obtain user commands If a valid TSO command was entered, TMP determines whether an operation is to be performed or a command processor is to be attached. Operations include displaying second level messages (help) or the time and performing no-ops (carriage return).

For command processors, TMP determines if the request will execute as a problem state or as APF authorized program. Both types of command processors are attached as a task under TMP to perform the TSO command. However, authorized programs become the only dispatchable unit of work for that address space. After the command processor completes, TMP determines if control was returned normally or if ABEND occurred.

When ABEND occurs, recovery is attempted by passing control to the TMP ABEND routine to determine where the error occurred. If the error occurred in a command processor, control is passed to its recovery routine or if none exists then to TMP. If this routine fails, then TMP regains control. Only when the routine successfully completes will the command processor continue. If the error occurred within TMP, then TMP attempts recovery, displaying the READY prompt if successful.

Another function of TMP is to monitor attention requests from the user. The requests occur during command processing and are handled for both problem state and APF ?uthorized command processing. When the attention key is entered, control is passed to the TMP exit routine and the READY prompt is displayed. For problem state processing, the exit routine scans for an operation. For example, the no-op operation causes the command to resume. If an attention occurred during APF authorized processing, then processing is terminated.

Although the IBM supplied TMP can be replaced by a installation specific program, any such program is not part of the evaluated system.

Command Processors

Users submit TSO/E commands to perform operations on data and data sets from a terminal. These commands are received and prepared for processing by TMP before a TSO/E command processor is attached. Command processors are problem state or APF authorized programs that perform TSO commands. Some command processors invoke system utilities, for example, a compiler.

TSO/E uses service routines to assist the command processors and TMP. These services are needed by all command processors and are executed as procedures. When the service routines are operating for a task, they perform at the same task level, as the task (see page 41,"Dispatcher"). The operations offered by services routines include: performing I/O operation to terminals, searching input buffers for TSO commands, and allocating or freeing data sets.

The User Attributes Data Set

TSO/E maintains a partitioned data set containing user attributes for their TSO/E session. This data set, SYS1.UADS regulates access to the system by maintaining information about the user identity, (TSO) password, account numbers, and procedure names.

The information contained in the SYS1.UADS data set can be copied (the entries will remain in UADS) to the RACF data base, when RACF is active. This process allows an installation to centralize TSO attributes with other data and eliminate the need to maintain entries in the SYS1.UADS data set. If this customization process is chosen (the administrator must have at least SPECIAL authority), the information contained in the UADS is converted into a command list (CLIST) and then placed into a data set. The administrator can specify all or a range of users from UADS, but all TSO attributes, for each user specified, will be placed in the data set.

The administrator will then have to issue an ADDUSER command for all specified users not previously defined to RACF. Once all users have an established RACF profile, the administrator instructs TSO to execute the CLIST. Execution of the CLIST will modify the user's TSO/E segment of the user's profile.

During identification, RACF will check the data base first to verify the user. However, if the userid is not found by RACF in data base, TSO/E will then check SYS1.UADS. If the userid is not contained in either one, the log on attempt fails. If the userid had been found in UADS, then the user would be logged-on, but unable to gain access to resources defined to RACF.

If RACF is deactivated, TSO/E reverts to the SYS1.UADS data set to check for authorized access to the system. Therefore, backup entries for system programmers should remain in UADS, but other user entries should be removed after conversion.

TSO/E Interface with VTAM

Data from TSO/E is passed to VTAM via the VTAM Terminal I/O Coordinator (VTIOC). VTIOC translates the I/O macros of TSO/E (e.g. TGET and TPUT) into VTAM SEND or RECEIVE macros. VTIOC then communicates with the terminal through the VTAM's Application Program Interface.

TPUT and TGET are macros which use an SVC to provide simple I/O. They are the basic macros used by TSO/E to transmit and receive a line of data. TPUT transmits data from an I/O routine to a terminal. TGET obtains data from a terminal and passes it to an I/O routine. The TGET and TPUT macros are the interface to the terminal via VTAM. The information transmitted via TPUT or TGET is called a TPUT message.

The GETLINE and PUTLINE macros are an interface to TSO/E routines which direct the request based on data set allocation. In the foreground, these allocations point to the terminal, so requests are routed to the terminal via the VTAM macros, TGET and TPUT. In the background and when the terminal operator has allocated data sets instead of the terminal for I/O, these macros route the requests to data sets using the QSAM access method. In this case, no TGET or TPUT requests are issued.

RESOURCE ACCESS CONTROL FACILITY

The Resource Access Control Facility (RACF) controls user access to resources by verifying user identities, authorizing user access to resources, and recording and reporting events via System Management Facilities (SMF). More specifically, RACF controls access to DASD data sets, tape volumes, and terminals. Access controls for all objects are described on page 96, "DISCRETIONARY ACCESS CONTROL". The information that RACF uses to control user access to resources is contained in profiles; these profiles are stored in the RACF data base. The RACF Manager routines handle all input/output to the RACF data base.

RACF Interface to MVS

RACF executes in the address space of its invoker. The default RACF installation places the RACF modules in SYS1.LINKLIB and SYS1.LPALIB. The RACF commands, RACF database initialization program, data security monitor, and RACF utilities must reside in an APF-authorized library. The RACF manager, RACF SVC processing routines, and RACF related exit routines run in key 0 and supervisor state, and reside in the link pack area (LPA, FLPA, or MLPA). The system uses seven macro instructions that interact with RACF, five of which issue SVCs. These macros may be called by authorized programs directly, or through SAF (see page 46, "System Authorization Facility"). When the instructions are issued through SAF, the MVS router passes control to the RACF Router. The RACF Router determines whether or not to call RACF for a particular request, and sets appropriate return and reason codes based on RACF processing.

Macros that issue SVCs:

RACINIT

> TSO/E LOGON and batch job initiators issue the RACINIT macro instruction to request that RACF verify the identity of the user attempting to enter the system. The RACINIT macro expands to issue SVC 131. If the user is RACF-defined, the RACF module that receives control verifies that the user's password, supplied group name (if any), and terminal authorization (if a TSO/E logon) are valid. If the user supplied the name of an application program, it also checks to see if the user has authorization to the application. If the user is authorized to enter the system, an Accessor Environment Element (ACEE) is built for the user. An ACEE is a control block specifying a user's authorizations throughout the life of the job. The ACEE resides in LSQA.

RACHECK

The resource managers issue the RACHECK macro instruction to determine if a user has authority to access a RACF-protected resource. The RACHECK macro expands to issue SVC 130. RACF verifies that the user is authorized to access the resource. RACHECK may also be issued by non-APF-authorized programs.

RACDEF

This macro instruction is issued by data management to define an entity (data set) to RACF or to change or delete the entity's RACF description. The RACDEF macro expandst issue SVC 133. If the user has the automatic data set protection (ADSP) or if the user specified PROTECT=YES JCL option (see page 84, "RACF Options"), data management invokes discrete profile processing. Otherwise, the resource managers apply generic profile processing (see page 83, "Resource Profiles").

RACLIST

RACLIST is issued by resource managers requiring very high performance checking to request that in-storage profiles (see page 81, "Profiles") be constructed for resources defined by a given class descriptor. The RACLIST macro expands to issue SVC 132. RACF builds an in-storage profile for a resource from the information obtained from the resource profile and from all resource groups that define the resource. RACF uses RACLIST to build in-storage profiles for all resources in a given class or for a specified list of resources from a given class (see page 83, "Resource Profiles").

RACXTRT

The RACXTRT macro instruction is issued to either retrieve a field, replace a field, or encrypt a password or other data from a user's profile. The RACXTRT macro expands to issue SVC 132. RACF obtains a work area for the user profile fields. If RACXTRT specified TYPE=EXTRACT, the address of the work area is passed back to the caller. If RACXTRT specified TYPE=ENCRYPT, the password text passed to the module is encrypted and the result is returned to the area that held the clear text.

Macros that don't issue SVCs:

FRACHECK

The RACF resource commands RDEFINE and RALTER issue the FRACHECK macro instruction to determine if a user has the authority to access a RACF protected resource. FRACHECK verifies access authority for only those resources whose profiles have been brought into memory by the RACLIST macro. If the profile isn't in memory, return code 4, resource or class name not found, is returned. The FRACHECK macro is not used by any evaluated software other than RACF.

RACSTAT

The resource managers issue the RACSTAT macro instruction to determine the status of RACF (active/inactive) and the status of a given class (active/inactive).

RACF Data Base

RACF maintains profiles in the RACF data base. The RACF data base holds all RACF access control information. RACF uses the data base each time a RACF defined user enters the system, each time a user wants to define access to a RACF-protected resource, and when a user accesses a RACF-protected resource (except through FRACHECK).

The RACF data base is a contiguous non-VSAM data set that resides on a DASD volume. It is made up of 1K blocks and is cataloged. The RACF manager addresses these blocks by relative byte addresses (RBAs). The RACF manager uses EXCPVR to read or write to a data base. When the system is IPLed, MVS opens and allocates the data base and updates the RACF control blocks with the physical locatior of the data base on the volume. The data set is then closed. To reduce device contention and to minimize the number of resources made unavailable by the loss of one device, the RACF data base may be divided into multiple physical data bases (up to 255) and spread across several devices. The RACF data base remains one logical data base. In the case of multiple physical data bases, the master scheduler initialization routine, at IPL time, constructs an internal RACF control block - the data base descriptor table. The data base descriptor table resides in common storage area (CSA) or extended common storage area (ECSA). The RACF manager uses this table to maintain and process the data bases.

RACF data bases consist of the following types of records: header blocks, block availability mask (BAM) blocks, index blocks, templates, and profiles. A description of each type of record follows.

The header or index control block (ICB) is the first block in a RACF data base and provides a general description of the data base. It contains information such as the total number of BAM blocks in the data base and the tape and DASD volume protection options. The ICB has a relative byte address of 0. RACF uses the ICB to locate the other blocks in a RACF data base. Each RACF data base has an ICB, but RACF uses only the ICB for the primary data set whcn determining the setting of options.

The BAM blocks determine the availability of all the blocks in a RACF data base. Each 1K BAM block contains header information followed by block masks. Each bit in these block masks corresponds to a 256 byte segment within a RACF data base block.

Index blocks are used to locate profiles. RACF uses a multiple level index to locate profiles in the RACF data base. All index searches begin with the highest level index block, whose RBA is contained in the ICB. At every level but the lowest, the first entry in a block that is equal to or alphabetically greater than the requested profile name is used to reach the next level index block. If no entry is greater than or equal to the profile name, the index search continues with the RBA pointed to by the last index entry in the block being searched.

RACF supplies a template for each type of profile (group, user, connect, data set, and general resource) and five unused templates that are reserved for future use. The templates contain a 14 byte definition for each field in the profile. This definition contains the field name, a set of five flags, and the field length. Each template also contains a number that corresponds to the type of profile it is describing.

Profiles contain descriptions of the attributes and authorities for every entity defined to RACF. The number in the entry type field identifies the type of profile and corresponds to the number of the template that maps this type of profile. Profiles will be further discussed on page 81, "Profiles".

RACF Manager

The RACF manager handles I/O to the RACF data base on behalf of the RACF commands and system macro instructions by using the EXCPVR (execute channel program virtual = real) instruction. The RACF manager also performs serialization and maintains the index structure and space allocation on the RACF data base. The RACF system SVCs (RACHECK, RACINIT, RACDEF, and RACLIST) branch directly to the RACF manager. The RACF commands interact with the RACF manager via SVC 132. The RACF manager processes nine requests: add a profile, alter a profile, alter a profile in place (size of the profile does not change), delete profile, delete a member of a tape volume set from a TAPEVOL resource class profile, locate a profile, find next profile of a given type, find next profile with the same first qualifier, and rename a profile.

Recovery

RACF allows the identification of backup RACF data bases that may be used in case of failure of the primary RACF data base. The backup data bases are allocated at the same time the primary data base is allocated. There are three backup options: all updates duplicated on the backup data base, all updates except for statistics duplicated on the backup data base, or no updates duplicated on the backup data base.

RACF data base recovery consists of two parts: the dynamic maintenance of a backup copy of the primary data base and the use of the RVARY command. The RVARY command is used to switch to the backup data base or to deactivate a specific data base to perform maintenance. When the RVARY command is issued, the operator must examine the userid to ensure that the issuer has the proper authority to enter the command. If so, the operator issues the password (if one has been defined using the SETROPTS command) or enters YES to allow RVARY to complete.

If all RACF data bases are deactivated, failsoft processing is in effect. For users that are already logged on, RACF uses whatever in-memory tables are still valid. If the user requests a profile that is not in a valid internal table, RACF prompts the operator to approve the request. If not already logged on, the only users that may log on are those who have userids in *SYS1.UADS* and know their UADS password. RACF then requests the operator's concurrence each time the user requests access to a general resource or to a data set that does not start with the user's ID.

RACF Users

A RACF user is identified by an alphanumeric userid. A RACF group is a collection of users having common access requirements. A RACF group is identified by an alphanumeric groupid. Users must be connected to one or more groups. For further discussion of userids and groupids and their attributes and authorities, see page 89, "IDENTIFICATION AND AUTHENTICATION".

A user's level of authorization is determined by a combination of four variables (for a description of how these variables are used to make access control decisions see page 96, "DISCRETIONARY ACCESS CONTROL"):

1. The user's attributes. The security administrator can assign attributes to eachRACF defined user. Attributes determine the privileges and restrictions a user has on the system. Attributes are classified as either user-level attributes or group-level attributes.

2. The user's group authorities. The security administrator or group administrator can assign a group authority to each user of a group.

3. The security classification associated with the user and the resources. Each user and each resource can have a security classification associated with it. The security classification consists of one ormore security categories and/or a security level. When a user requests access to a resource that has a security classification, RACF first compares the security level of the user with the security level of the resource. If the user's security level is sufficient to access the resource, RACF compares the list of security categories associated with the user with the list of security categories associated with the resource. If RACF finds any security category associated with the resource and not associated with the user, RACF denies the request. Although RACF provides minimal mandatory access control, it does not meet the Criteria Mandatory Access Control requirements.

4. The resource access authorities associated with the various data sets and general resources. Resource access authorities determine to what extent the specified user or group can use the resource. Each resource also has a universal access authority (UACC) associated with it. It is the default access authority for the resource. It allows any user or group to access the resource with this authority unless RACF has already permitted or denied the user or group access to the resource. For further discussion of resource access authorities see page 96, "DISCRETIONARY ACCESS CONTROL".

Profiles

The information that RACF uses to control access to protected resources is contained in RACF profiles. Each profile is owned by a user or group. By default, the owner of a profile is the user who creates it. There are five types of profiles: user, group, connect, and two resource profiles (data set and general resource).

User Profiles

A user profile defines an individual user. When a user is defined to RACF, a user profile is created in the RACF data base. A user profile consists of a RACF segment and an optional TSO/E segment. Each segment consists of fields. The RACF segment of a user profile contains basic information needed to identify a user to RACF such as userid, name, authority, password, and UACC. The TSO/E segment contains the TSO/E attributes for the user such as account number, default logon procedure, and default region size.

A user profile is created by using the ADDUSER command. The issuer of this command must have the SPECIAL attribute or must have the CLAUTH attribute for the USER class and meet one of the following conditions: be the owner of the default group specified in the command, have JOIN authority in the default group specified in the command, or the default group specified in the command must be within the scope of a group (for more information on group scope see page 89, "Groups") in which the issuer has the group-SPECIAL attribute. The owner of the user profile is specified by using the OWNER parameter of the ADDUSER command. If no owner is specified, the user creating the profile is defined as the owner.

Group Profiles

A group profile defines a group of users. When a group is defined to RACF, a group profile is created in the RACF data base. A group profile consists of one segment called the RACF segment. The RACF segment of a group profile contains basic information needed to define a group to RACF, such as the group name and the owner of the group's profile.

The ADDGROUP command is used to create a group profile. To use the ADDGROUP command, the user must have the SPECIAL attribute, or have the group-SPECIAL attribute within the superior (for a discussion of superior groups, see page 89 "Groups") group, or be the owner of the superior group, or have JOIN authority in the superior group. The OWNER parameter of this command specifies the group or user to be assigned as the owner of the new group. If the OWNER parameter is not used, the user creating the new group profile is defined as the owner.

Connect Profiles

Connect profiles reflect the relationships between a user and one or more groups to which the user belongs. The connect profile contains the owner of the profile, the user attributes effective within the scope of the group, and other information about the group. A connect profile is created automatically whenever a new user is defined to RACF and connected to a default group or whenever a previously-defined RACF user is connected to an existing RACF group.

A connect profile may also be created by using the CONNECT command. To use this command, a user must have the SPECIAL attribute, or have the group-SPECIAL attribute in the group being connected to, or be the owner of the group, or have JOIN or CONNECT authority in the group. The OWNER parameter may be used to specify the user or group to be assigned as the owner of the connect profile. The default owner is the user creating the profile.

Resource Profiles

Resource profiles can be broken down into data set profiles and general resource profiles. Data set profiles define access to DASD and tape data sets while general resource profiles define access to any resource other than a user, group, or data set (e.g., terminal). Resource profiles contain information such as the resource name and owner, a list of users who may use a resource and how they may use it, the default level of access authority allowed for all users not listed in the access list (i.e., the UACC), and auditing information.

Data set and general resource profiles can be further divided into discrete and generic profiles. A discrete profile defines the protection of a single object. That is, there is a single unique object protected by the profile. A generic profile defines the protection of a group of objects. The scope of protection provided by a generic profile encompasses more than one object, usually having similar names.

Generic Profile Name	Matching Names
ABC.EFG.*	ABC.EFG.H.IJKLM
	ABC.EFG.HIJKL
	ABC.EFG.HIJ.KLM.NOP

General resource profiles are used to protect objects within a class of resources, other than class DATASET, (e.g., DASD volumes or tape volumes) that have been defined to RACF. A class is a collection of RACF entities with similar characteristics, for example USER or DATASET. IBM predefines several general resource classes which include tape volumes, DASD volumes, and load modules. The control information for general resource classes is contained in the class descriptor table. The control information includes the resource class name and the syntax rules for the resource names within the class. A site can define more resource classes which could then be protected by RACF. The RDEFINE command is used to define resources belonging to classes that are specified in the class descriptor table. To use the RDEFINE command, a user must have the SPECIAL attribute or CLAUTH to the class.

Audit

RACF has the ability to audit events where user-resource interaction has been attempted. The actual access activities or variances from the expected use may be recorded. RACF audits by writing records to a System Management Facilities (SMF) data set. The RACF report writer is used to extract pertinent SMF records. For more information see page 107, "AUDITING".

RACF Commands

The RACF commands are used to create, alter, list, or delete profiles and to define system-wide options. To successfully issue a command, a user must be defined to RACF with a sufficient level of authority. RACF commands are entered during a TSO/E session by entering the commands directly or by using the RACF Interactive System Productivity Facility (ISPF) panels. All RACF command functions, except RVARY and RACFRW, have ISPF entry panels (menu) and associated help panels. Some commands that may be used for user definition tasks include ADDUSER, ALTUSER, CONNECT, DELUSER, and LISTUSER. The SETROPTS command is used to set system-wide RACF options such as enable or disable the global access checking facility and activate and control the scope of erase-on-scratch processing. The RVARY command is used to deactivate and reactivate the RACF function and to switch from using the primary RACF data base to the alternate RACF data base.

RACF Options

The evaluated system has three RACF installation options which must be active: PROTECTALL, JES BATCHALLRACF, and ERASE(ALL). PROTECTALL causes RACF to allow the creation of data sets only if the data set will be protected by a profile. The JES BATCHALLRACF option specifies that JES is to test for the presence of a userid and a password on the job statement, or JES propagated RACF identification information for all batch jobs. ERASE(ALL) specifies that all DASD data sets, including temporary data set, are physically erased upon deletion. For additional information on ERASE(ALL), see page 120, "Object Reuse". PROTECTALL, JES BATCHALLRACF, and ERASE(ALL) options are set using the RACF SETROPTS command. Another option, generic profile checking, is recommended but not required in the evaluated configuration. If generic profile checking is active, RACF will search for a generic profile when a discrete profile cannot be found.

PROTECTED RESOURCES

SUBJECTS

Subjects in MVS/XA with RACF are address spaces performing user and system functions. There are five types of subjects: console operators, started tasks, system services, TSO/E users, and batch jobs.

Console operators (users) are tasks providing an interface between the system operator terminal(s) and the system. An operational system has at least one operator console terminal and an operator. Console users are the only subjects sharing a common address space; eac'n console user has a task within one communication address space.

Started tasks are tasks initiated by explicit commands taken from a system start-up data set or received from an operator. System components such as JES, VTAM, and TCAS are examples of started tasks. Started tasks commonly execute until either the system is stopped or the operator deletes them. Each started task is assigned its own address space for execution.

System services are tasks executing the system code in order to accomplish various user-related functions. Such tasks execute within a user's address space but utilize key protection to prevent users from modifying these tasks. The Region Control Task (RCT - see page 36, "Address Space Creation") is an example of such a system service.

TSO/E tasks represent logged-in TSO/E (interactive) users. Each TSO/E user is assigned its own address space. TSO/E users are provided with ways of sharing data, but unless they are authorized users, they cannot access and modify data in other address spaces. The users and their RACF-built ACEEs are collectively maintained throughout their interactive sessions.

Batch jobs can be created by each of the previously described subjects. Batch jobs are created by submitting jobs to a JES for background processing. Executing batch jobs can, via the internal reader, submit other batch jobs. A batch job executes in the address space of an initiator processing that job (see page 36, "Address Space Creation").

OBJECTS

MVS/XA with RACF provides users with direct or indirect access to ten kinds of objects: direct access storage device (DASD) data sets, spool data sets, temporary data sets, tape volumes, address spaces, volume tables of contents (VTOCs), TPUT messages, catalogs, VTAM logical units, and terminals. This section provides a definition of these objects.

DASD Data Sets

DASD data sets (disk files) are the basic containers in which information is stored in MVS/XA. With respect to security, there are two types of data sets in MVS/XA: virtual storage access method (VSAM) data sets, and non-VSAM data sets. Users may be authorized to control the contents and the access to both of these types of data sets. For further reading see page 62, "Data Sets".

Spool Data Sets

JES2 is the system function that spools and schedules input and output data streams. These data streams are designated as SYSIN and SYSOUT. SYSIN and SYSOUT are implemented as data sets and they are opened and closed in the same manner as any other data set processed on a unit record device. Each job may be associated with many such spool data sets. For further information see page 56, "Spool Data Sets".

Temporary Data Sets

Temporary data sets are used to store data for the duration of the current job. The Virtual Input/Output (VIO) operation provides one temporary data set capability through the use of the system paging data sets; VIO uses the system paging routines to transfer data to and from paging data sets. Other types of temporary data sets may be implemented as VSAM and non-VSAM data sets. In these cases the data sets are system-owned, not cataloged, and their names are generated automatically. In all cases temporary data sets exist only for the duration of a job they are associated with, and they cannot be shared among users. For additional information see page 35, "Virtual Input/Output" and page 62, "Data Sets".

Tape Volumes

A tape volume represents a collection of one or more data sets stored on a magnetic tape. Users may be authorized to control the contents and the access to tape volumes. For further discussion see page 60, "Magnetic Tape Volumes".

Address Spaces

An address space contains information pertaining to execution of related tasks. Address spaces may be shared among users. This is accomplished using cross memory services. Users must be authorized to utilize this service. For a detailed description see page 30, "Cross Memory Services".

Volume Tables of Contents

The volume tables of contents (VTOCs), described on, page 61, "Volume Table of Contents", are disk directories containing information about allocation of space on each disk. Users indirectly affect the contents of VTOCs through allocation and deallocation of data sets.

TPUT Messages

Terminal users may communicate with one another using TPUT (terminal write) messages. Users prepare message text, but once that text is sent, no further user control over it is possible. User may choose not to receive TPUT message from other users. For an additional information see page 73, "TIME SHARING OPTION/EXTENSIONS".

Catalogs

Catalogs are data set and catalog directories. Users may add and delete entries for the data sets they own. Since catalogs are implemented as VSAM data sets, conventional protection mechanisms are applied to these data sets to prevent unauthorized access. Users utilize system-provided functions to modify entries in catalogs. For a detailed description see page 61, "Catalogs".

VTAM Logical Units

A logical unit provides the means by which a user or an I/O mechanism gains access to the system. ACF/VTAM system service is itself a logical unit connected with a logical unit of a terminal. This connection is established by the system administrator typically during the system start-up. A non-privileged user cannot modify this connection. For further discussion of sessions between logical units see page 68, "ACF/VIRTUAL TELECOMMUNICATIONS ACCESS METHOD".

Terminals

After defining a terminal's logical unit to ACF/VTAM, the terminal device identifiers can then be used to specify which terminal devices users may log onto. A system administrator may restrict users to logging in only through specific terminals.

This page intentionally left blank.

PROTECTION MECHANISMS

IDENTIFICATION AND AUTHENTICATION

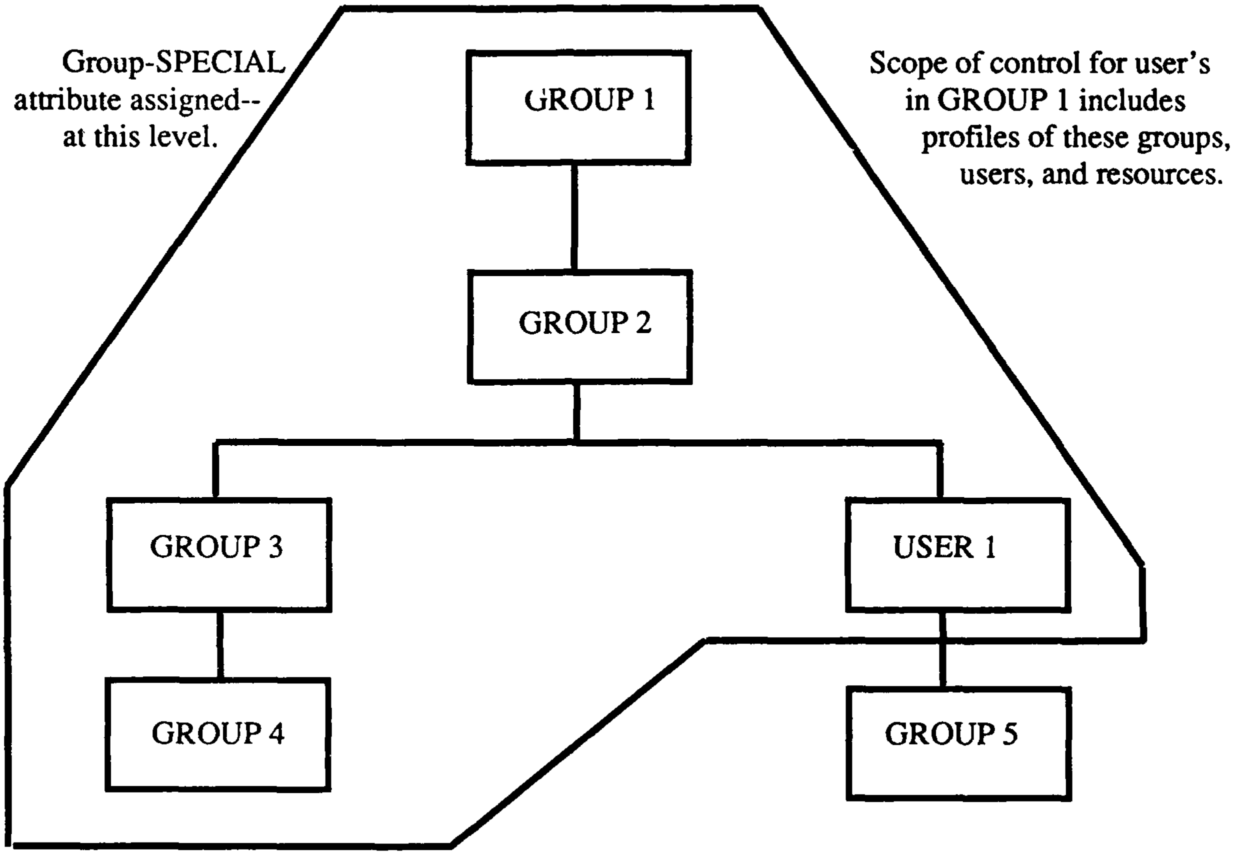
MVS/XA with RACF provides identification and authentication through the use of RACF. Identification and authentication is applied to all subjects except console operators. Any subject identified and verified by RACF is a RACF user. Once a user is verified an ACEE is created. A RACF user must also belong to at least one group. A RACF user is placed in a default group unless another group is specified at logon. A user can only be active in one group at a time, however, a user may have the combined authority of all the groups the user belongs to. To change the active group a user must logoff and relogon specifying the new group.

Groups

In RACF, a group is a set of users with the same access requirements. Groups are flexible and can be structured to reflect the organization of logical entities like departments or projects. When groups own other groups the owning group is called the superior group. The privileges and restrictions a user has within a group apply only to the resources within the scope of the group. Resources within the scope of the group are:

- resources owned by the group,
- resources owned by users who are owned by the group,
- resources owned by subgroups that are owned by the group.

The scope of control of a group percolates from a group to its subgroups. The percolation stops when a subgroup is owned by a user. (See figure 6.)



The direction of authorization is from top to bottom.
Figure 6

Figure 6 shows the scope of control of an attribute assigned at the group-level. GROUP1 owns GROUP2, GROUP2 owns GROUP3 and USER1, and so on. Attributes assigned to GROUP1 apply to anything owned by GROUP1, and if GROUP1 owns a group those attributes apply to what that group owns as well, and so on. In the figure a user is connected to GROUP1 with the group-SPECIAL attribute. This allows that user to use the group-SPECIAL privileges within the encircled area. In other words, the connected user (and any user with the group-SPECIAL attribute in GROUP1) can access the profiles and resources owned by GROUP1, the profiles owned by GROUP2 (GROUP2 is owned by GROUP1), the profiles owned by GROUP3 (GROUP3 is owned by GROUP2), the profiles owned by GROUP4 (GROUP4 is owned by GROUP3), and the profiles owned by USER1. The connected user cannot access the profiles or resources in GROUP5 because GROUP5 is owned by a user. This is also true for a user owned by a user (for example, if USER1 owned a user).

Attributes and Authorities

A subject's authorities are contained in its user, group, and connect profiles. When a privileged user (a user who has the SPECIAL attribute, the CLAUTH attribute for class USER and the JOIN authority to a group, or the group-SPECIAL attribute, see descriptions of these below) defines users and groups to RACF, the information is placed in profiles. The attributes contained in a profile define the user's or group's privileges and restrictions. The user's profile also contains group authorities, giving a user authorities in a group. A connect profile is created automatically when a new user is defined to RACF and connected to a default group or when an existing user is connected to an existing RACF group with the CONNECT command.

The user attributes are SPECIAL, AUDITOR, OPERATIONS, CLAUTH, GRPACC, ADSP, and REVOKE. The attributes apply regardless of what group the user is in. The group attributes are group-SPECIAL, group-AUDITOR, and group-OPERATIONS, which apply only to the group (and the scope of control within that group) the user has the group attributes for. There are also group authorities assigned by the group administrator to each user. The group authorities are USE, CREATE, CONNECT, and JOIN. A description of each attribute and authority follows.

Attributes

The SPECIAL attribute allows the user to issue all RACF commands and gives the user full control over the RACF profiles. This attribute can only be given by a user with the SPECIAL attribute. The group-SPECIAL attribute gives the user full control over the resources within the scope of a group only, i.e., the effects of RACF commands will only apply to resources within the scope of the group. A user with the SPECIAL attribute is usually designated as the security administrator.

The AUDITOR attribute gives the user the responsibility for auditing the security controls and the use of the system resources for the whole system. The user assigned the AUDITOR attribute can specify logging options on RACF commands, can list auditing options of profiles, and can control additional logging to the SMF data set. The user may also list the profile information available to the SPECIAL user. This attribute can only be assigned by a user with the SPECIAL attribute. The group-AUDITOR attribute restricts authority to the resources within the scope of the group.

The OPERATIONS attribute allows the user to perform maintenance functions like copying, reorganizing, cataloging, and scratching RACF-protected resources. The group-OPERATIONS attribute restricts authority to the resources within the scope of the group.

The CLAUTH (class-name authorization) attribute is given to users on a class-by-class basis and it cannot be assigned at the group level. A class is a collection of RACF entities with similar characteristics (see page 83, "Resource Profiles"). CLAUTH allows the user to define profiles in that class to RACF. The CLAUTH user may also add new users to RACF if the CLAUTH user is the owner of or has JOIN authority to a group. This group will become the new user's default group. The CLAUTH attribute also allows the user to define resources to be included in the assigned class.

The GRPACC (Group Access) attribute makes any group data set profiles the user defines to RACF automatically accessible to other users in the group if the user defining the profile is a member of that group. If assigned to the user, this attribute applies to all groups the user is a member of, and if assigned at the group level, the attribute applies only to that group.

The ADSP (Automatic Data Set Protection) attribute causes every data set created by the user to have a discrete profile automatically created. If this attribute is assigned at the group level, ADSP is only in effect if tne user is within the group.

The REVOKE attribute provides the capability to prevent a RACF user from entering the system. If assigned at the group level, the user cannot enter the system by connecting to that group or access the resources of that group. Using RACF commands, a future time can be set for REVOKE to occur or for REVOKE to be removed. The owner of a user's profile or the system administrator can assign the REVOKE attribute. A user with the REVOKE attribute can also specify how many consecutive logon attempts are permitted by RACF before the userid is revoked.

Authorities

The group authority USE allows a user to access data sets within the group and to create RACF protected data sets.

The group authority CREATE allows a user to RACF-protect and control access to data sets within the scope of the group. The CREATE authority includes the privileges of the USE authority.

The CONNECT group authority includes the privileges of the USE and CREATE authorities. It also allows the user to connect users to a group, and the user may assign USE, CREATE or CONNECT group authorities to users in that group.

A user with the JOIN group authority can define new users (provided the user also has the CLAUTH attribute for the USER class) and groups to RACF and give those new users any level of group authority. The JOIN group authority allows a user possessing that authority to create new groups. The newly created group will become a subgroup of the group the user has JOIN authority to. The JOIN authority includes the privileges of the USE, CREATE and CONNECT authorities.

Userid, Password, and Groupid

RACF requires a userid and password from a user attempting to logon. There may also be further checks to limit logon such as terminalid, time of day, or day of week. A user may be limited to enter the system on certain days of the week and during certain hours of the day. A user can also be restricted to use specific terminals on certain days of the week and during certain hours each day if the terminalid is used.

The userid must be one to seven characters in length and it must start with an alphabetic or a national character ($, @, and #). The groupid, like the userid, must start with an alphabetic or national character. The groupid identifies a group to RACF. Userids and groupids must be unique. MVS/XA with RACF maintains and protects userids through all the steps of job/task execution.

The RACF password can be from one to eight characters in length and is chosen by the user at the first logon. The initial password is chosen by the authorized person that added the user to the system. The password can have up to eight password syntax rules specified by a user with the SPECIAL attribute. The rules allow the privileged user to control minimum and maximum length of passwords and the character content of site-selected positions in the passwords. This authorized user can also set the maximum password change interval for all users, but unprivileged users can make their own interval shorter. RACF has the capability to store a number of old passwords to force a user to choose different passwords at each change. The RACF password data are stored in the RACF data base in either a masked or encrypted form. By default an IBM supplied exit routine in RACF masks the password data. An installation can encrypt the password data by removing or modifying the supplied exit routine in RACF. A software implementation of the DES encryption algorithm is provided, but sites can use their own algorithms.

Mapping Subjects to Userids

Each type of subject in MVS/XA with RACF is initially associated with its userid in a different way. This section describes how a subject is associated with a userid.

Batch Jobs

JES propagates the current RACF userid when an already validated RACF user submits a batch job to JES via JES internal readers. The jobs are marked as password validated so that password validation is not performed. When the initiator processes the job, the propagated userid and the user's default group are used by RACF to create the ACEE.

Jobs submitted by a RACF/TSO/E user are automatically identified with that user and the default RACF group of that user. In this case a password is not required in the JCL JOB statement. If a user wishes to submit a job in a group other than the default group, the user must specify the userid, password and groupid when the job is submitted. However, if a job is submitted in a user's current connect group, only the groupid is required on the JOB statement. If a TSO user submits a job for another user, the userid and password (groupid is optional) must be present on the JOB statement.

To prevent unauthorized users from running batch jobs, a site can specify that RACINIT be invoked for all batch jobs. This is accomplished with the RACF SETROPTS command option JES(BATCHALLRACF). The BATCHALLRACF indicator sets a flag that JES tests. JES fails any job before the conversion phase that does not have propagated user identification or a userid and password on the JOB statement. This action is mandatory to meet class C2 requirements for identification and authorization.

Started Tasks

When an operator issues the START command, an address space is createc for a started task. The START command includes the name of the procedure to be executed. Since no identification or authentication information is included with the START command, a userid/groupid replacement table is scanned for the procedure name. The userid/groupid replacement table contains a procedure's name, its associated userid and groupid, and flags. The table is created during RACF installation, and it resides in the link pack area. If either userid or groupid is found for the procedure's name, this information is placed in the ACEE and is used by RACINIT to identify the user (see page 76, "RACF Interface to MVS"). If the procedure's name is not found, a generic entry is looked for. If no entry is found the default userid (*) and groupid (*) are placed in the ACEE and null authority is given to the started task. With the default userid and groupid the started task can access RACF protected resources only if the universal access authority for the resource allows the access.

TSO/E Logon

A TSO/E logon is initiated with the TSO LOGON command. When a user logs onto TSO/E, TSO/E checks the TSO segment in the user's RACF profile for the user's authority to use TSO/E resources. If a user doesn't have a TSO segment, then TSO/E checks SYS1.UADS for the information to build a session for the user. TSO/E then issues the RACINIT SVC and passes the userid, groupid, and password information to RACF. If the user is identified and verified, RACF builds an ACEE for the user.

In the above three cases (batch jobs, started tasks and TSO/E logon) if identification or verification fails, the ACEE storage is freed and a non-zero return code is returned to the caller of the RACINIT SVC.

Console Operator

Console operators issue commands from a single address space called the communication task address space. This address space is a non-swappable system address space created during system initialization. It transfers messages from user programs and system routines to the operators at the consoles. The physical consoles are defined to the system at system initialization. Console operators do not log onto the system; they are physically granted access to the consoles since they are trusted individuals.

DISCRETIONARY ACCESS CONTROL

This section discusses the mechanisms MVS/XA with RACF uses to provide discretionary access control. Checks that are made in determining a user's access to protected objects are identified. This section also addresses the modes of access to protected objects.

Discretionary access control is enforced in the system by both MVS/XA and the RACF program product (see page 76, "RACF Interface to MVS"). MVS/XA provides the abstraction and separation of subjects. RACF maps every subject to a userid and groupid (see page 93, "Mapping Subjects to Userids") to identify the user attempting to gain access to protected objects. Some of the objects described on page 86, "OBJECTS" (i.e., DASD data sets, tape volumes, and terminals), are protected by RACF profiles. Other objects (e.g., spool and temporary data sets) are protected by MVS/XA and its subsystems.

RACF Protected Objects

MVS/XA and RACF protect objects through the use of profiles stored in the RACF data base. A profile can be used to define the access a user has to an DASD data sets, tape volumes and terminals. The contents of a profile are identified on page 83, "Resource Profiles". Profiles that protect tape volumes and terminals are called general resource profiles, while DASD data set profiles are called data set profiles. One area in which data set profiles differ from general resource profiles is the access attribute checks that are performed. This difference is described in the following section. Both general resource and data set profiles can be either discrete or generic (see page 83, "Resource Profiles").

With generic profile checking active, RACF searches for profiles that match a protected object by first searching its data base for a discrete profile. If a discrete profile does not exist, RACF searches for a generic profile in order from most specific to least specific. If a profile is found it is used to determine access to the object. If no profile exists for an object then access is denied.

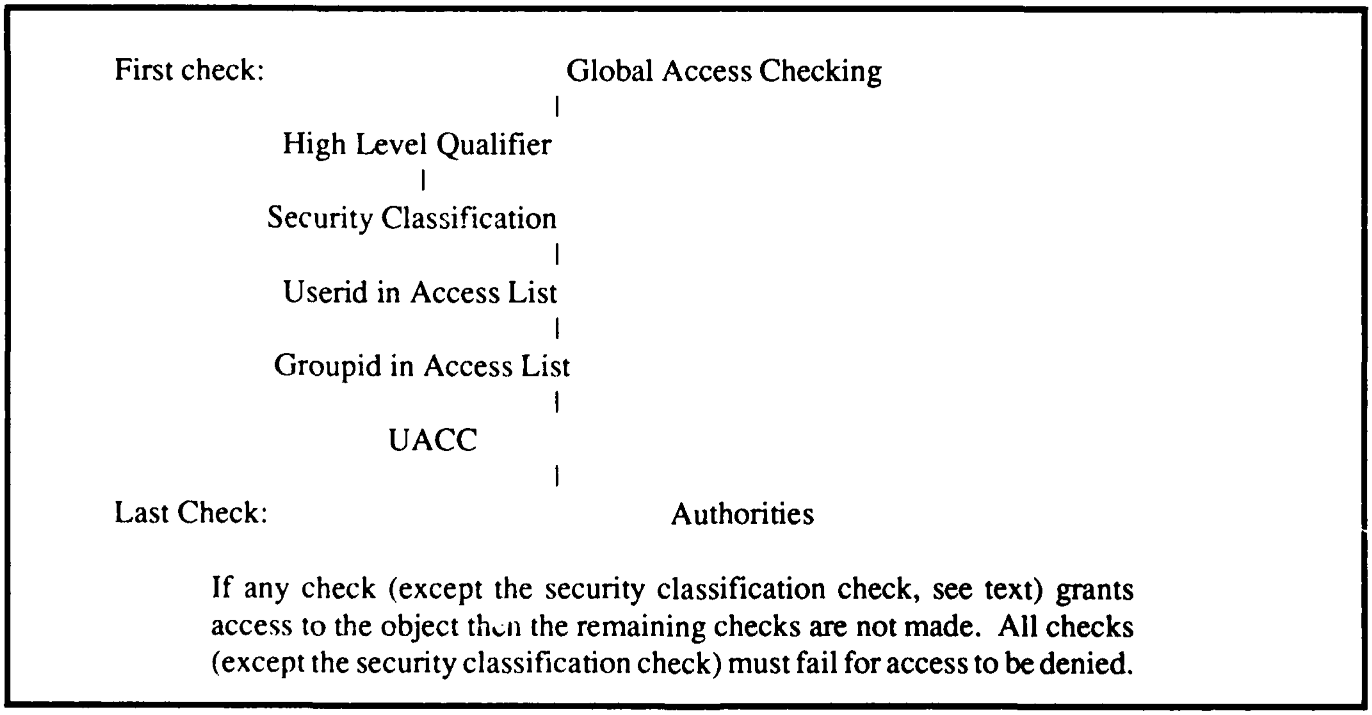
In addition to profiles, RACF provides another method of granting access to objects, called global access checking. Global access checking is a system-wide option, and does not perform a check against an objects profile. If global access checking is active (optional in the evaluated system), RACF searches the global access table for a match between the desired object and access mode and entries within the global access table. If a match permitting the requested access mode is found, access permission is granted. If a match is not found, or the match does not permit the requested access mode, or global access checking is inactive, RACF searches for a profile. Global access checking can only allow access to an object; it can never cause access to be denied. Access granted because of global access checking is not audited by RACF (see page 108, "Logging").

Profile Inspection

Once a profile has been found, RACF performs slightly differently for data sets than for general resources. Figures 7 and 8 describe the access checks performed for data sets and non-data sets, respectively. For data sets, RACF will check to see if the high level qualifier of the data set matches the user accessing the data set, and if so, it will automatically grant ALTER (see page 98, "Resource Access Authorities") access to that data set. If the high level qualifier does not match the userid, then RACF will examine the security classification of the data set. Should the user not have a sufficient security classification to access the data set RACF denies access. A successful security classification check does not grant access but rather allows further checking to continue. The owner of a resource always has the ability to change the profile (and possibly give himself or herself access authority).

When a profile has been found for a general resource, RACF does not check for ownership of the resource, but begins with a check of the security classification of the resource. (Please note the difference between figures 7 and 8.)

DATA SET ACCESS CHECKS



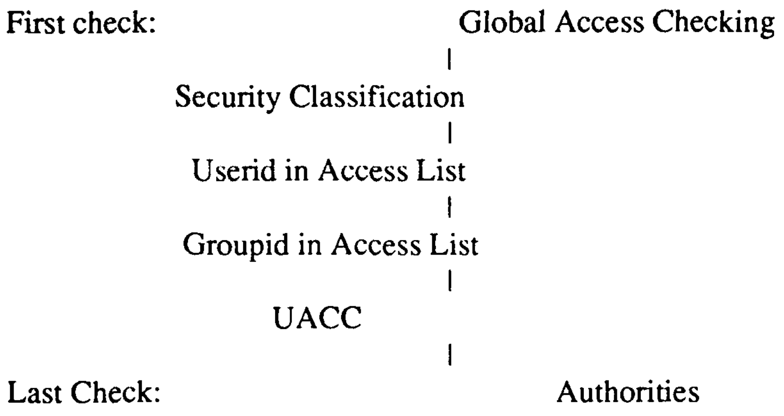
First check: Global Access Checking
High Level Qualifier
Security Classification
Userid in Access List
Groupid in Access List
UACC
Last Check: Authorities

If any check (except the security classification check, see text) grants access to the object then the remaining checks are not made. All checks (except the security classification check) must fail for access to be denied.

Figure 7.

The access list in the profile is checked next. If the userid or groupid is found in the access list, success depends upon the access mode requested and the access mode specified in the access list. If the userid or the groupid is not found in the access list, then the universal access authority (UACC) is checked. The UACC defines the mode of access permitted to users not explicitly named by the access list. If the requested access mode is not permitted by the UACC then attributes (i.e., OPERATIONS, group-OPERATIONS) are examined before access is determined.

NON-DATA SET ACCESS CHECKS



First check: Global Access Checking

Security Classification

Userid in Access List

Groupid in Access List

UACC

Last Check: Authorities

If any check (except the security classification check, see text) grants access to the object then the remaining checks are not made. All checks (except the security classification check) must fail for access to be denied.

Figure 8.

Resource Access Authorities

The system provides five predefined RACF access authorities (i.e., access modes) used when accessing profile-protected resources: NONE, READ, UPDATE, CONTROL, and ALTER.

NONE	This access authority does not permit a user or group to access the protected object.
READ	This access authority allows a user or a group, access to the resource for the purpose of reading the object only.
UPDATE	This access authority allows read and write access to the resource.

CONTROL This access authority varies depending upon the resource being protected and is described in the following discussion of each protected resource.

ALTER This access authority is different for discrete and generic profiles. If ALTER authority is provided by a discrete profile then a user is allowed to control the contents of all fields of the profile as well as to control the contents of the object. ALTER authority provided by a generic profile allows a user to control the contents and existence of the data set. Control over the contents of the profile requires that a user be the owner of the profile, have either the SPECIA or group-SPECIAL attribute, or a userid that matches the high level qualifier of the profile. A high level qualifier is the portion of a name (data set or profile) that appears before the first period, having a maximum of eight characters.

DASD Data Sets

Data set profiles protect both VSAM and non-VSAM data sets. MVS/XA through DFP provides a mechanism to protect data sets with passwords. If a MVS/XA data set password is used along with RACF, the password is ignored and access is calculated using the RACF profiles.

RACF interprets the access authorities of NONE, READ, UPDATE and ALTER as described on page 98, "Resource Access Authorities" for DASD data sets. The CONTROL access authority however is interpreted differently for VSAM and non-VSAM data sets. For non-VSAM data sets, the CONTROL access authority is equivalent to UPDATE. For VSAM data sets, the CONTROL access authority permits a user to access (for both read and write) a VSAM data set's control interval (see page 63, "VSAM Data Set Organization").

Tape Volumes

While RACF attempts to provide access to the level of individual data sets on a tape volume, it cannot guarantee enforcement of this policy for more than one data set per tape. RACF provides access controls to the granularity of a tape volume. That is, users granted access to a tape volume have access to all data sets on the tape volume.

Administrators enable RACF protection of tape volumes by making the TAPEVOL class active. Access checking is then performed whenever a tape volume is accessed (e.g., OPEN) The RACF command, SETROPTS CLASSACT(TAPEVOL), activates the TAPEVOL class. RACF protects only defined tape volumes with an IBM standard or ANSI label. A tape volume is defined in two ways:

1. Issuing the RACF RDEFINE command without the TVTOC operand.
2. Issuing the RACDEF macro and the JCL PROTECT operand.

If a tape volume is not defined to RACF access is granted based on password protection of data sets on a tape.

Only users with the CLAUTH attribute for the TAPEVOL class may define a tape volume or create a profile for a tape volume. For tape volumes, RACF interprets the access authorities of NONE, READ, and UPDATE as described on page 98, "Resource Access Authorities". RACF issues a message to the operator to remove the write-enable ring if a tape volume is to be read-only. The access authority of CONTROL is equivalent to the access authority of UPDATE for tape volumes. The access authority of ALTER allows a user to overwrite the tape label, and modify the tape volume profile.

If a data set spans multiple tape volumes it is called a tape volume set. RACF grants UPDATE access based upon the the profile of the first tape volume in the set. READ access is based on the profile of each tape volume in the set that is RACF-protected.

Terminals

If the TERMINAL class exists, RACF controls access to terminals. Profiles are used to specify which users, or groups, may use a terminal.

Two access authonties are valid for terminals, READ and NONE. Users with NONE access to a terminal may not access the system through that terminal.

Other Protected Objects

Objects not protected by RACF are protected by mechanisms other than profiles. The objects protected by MVS/XA with RACF that do not have profiles are spool data sets, temporary data sets, address spaces, volume table of contents, TPUT messages, catalogs and VTAM logical units. These objects can be grouped into non-sharable objects that are private to the object owner (e.g., a temporary data set) and system controlled objects (e.g., a volume table of contents).

Spool data sets, temporary data sets and address spaces are protected such that non-privileged subjects other than the object owner have no access to the object. Volume tables of contents, catalogs, TPUT messages, and VTAM logical units are system objects. A subject's access to these objects is determined by the subject's access to another object. For example, a subject with ALTER access to a DASD data set may delete that data set's VTOC entry when the DASD data set is deleted. The specific controls applicable to each object are explained below.

Spool Data Sets

Two spool volumes are maintained by JES, SYS1.HASPACE and SYS1.HACKPCKPT. JES provides per-job data sets (i.e., SYSIN, SYSOUT) whose information is stored within SYS1.HASPACE. The SYSIN and SYSOUT data set of one job cannot be access by another job. These data sets are not shared between jobs.

Temporary Data Sets

VIO temporary data sets are data sets that exist within an address space, and are destroyed when the address space terminates. Other types of temporary data sets may exist outside of their owner address space but they, too, are destroyed when their owning job terminates. Temporary data sets may only be shared among tasks within the same address space (i.e., within the same job).

Address Spaces

Address spaces are isolated from one another by means of segment and page tables. The system does not allow unprivileged users to share address spaces. Jobs within an address space may access information anywhere within that address space, however, all these jobs belong to the same user.

Volume Table of Contents

The VTOC is a system maintained data structure stored as a data set. RACF can be used to ensure that unprivileged users have only read access to a VTOC data set. However, unprivileged users may indirectly cause entries in a VTOC to be created or deleted during the creation or deletion of data sets. Only the system may write to VTOCs, and only through the use of APF authorized programs.

Catalogs

The master catalog and user catalogs are VSAM key-sequenced data sets. An unprivileged user's access to these catalogs is mediated by DFP. Unprivileged users may read a catalog; only system key, supervisor state or APF authorized programs may write to a catalog. Unprivileged users may indirectly write to a catalog through the creation and deletion of cataloged data sets. As described earlier (see page 96, "RACF Protected Objects"), RACF controls a user's access to data sets.

TPUT Messages

TPUT messages pass between a terminal and a VTAM application program only after a session has been established. VTAM ensures that data correctly passes between two LUs that are in session. Therefore, a VTAM application program may only send TPUT messages to the terminal that has opened a session with the address space containing the VTAM application program.

VTAM Logical Units

VTAM logical units are not directly controllable by users. Logical units can only be defined by a system administrator. A user, upon logon, initiates a session between the terminal's LU and a VTAM application program LU. This VTAM application program LU resides within an address space that will become the user's address space. Once this session is established, it is only terminated by a logout.

OBJECT REUSE

Reuse of the protected objects is handled by the RACF ERASE option, administrative practices, and by other software controls.

RACF ERASE Option

RACF maintains the ERASE option which directs the operation of deletion. This option is set using the Set RACF options (SETROPTS) command to specify the condition as to when data sets and catalog entries are to be erased or physically overwritten with zeros. The four possible arguments are outlined below:

ALL	All data sets are overwritten.
SECLEVEL	Data sets above a specified security level are overwritten
NOSECLEVEL	Data sets which have their profile erase indicator on are overwritten.
NOERASE	No data sets under RACF control are overwritten.

For the evaluated system, the ERASE parameter must be set to ALL, ERASE(ALL) so that the physical extents of DASD data sets are erased or overwritten with zeros at the time of deletion.

RACF does not erase any of the extents, but instead maintains the argument specified. The deletion is actually executed by the DFP DELETE utility which references the ERASE argument. This utility is called from by the DELETE command or by JCL instructions.

DASD Data Sets

The ERASE option affects the erasure of VSAM data sets and all non-VSAM data sets, including single and multiple volume data sets.

Catalogs

Catalogs are data sets. If a catalog is deleted, the ERASE option will cause the catalog to be overwritten.

Catalog entries are overwritten when the data set is deleted using the DELETE command or when it is specified as a cataloged data set in the JCL instructions. However, if the volume and serial number are specified outright in the JCL instruction, then MVS/XA with RACF has no way of knowing that the data set was cataloged. The catalog entry will then be left unaltered.

Controlling Reuse of Other Objects

The following objects are not controlled by the ERASE option: VTOC, tape volumes, address spaces, spool data sets, TPUT messages, VTAM Logical Units (LU), and system terminals. Data reuse for these objects is handled by administrative practices or by the other software control, as detailed below.

Volume Table of Contents

Entries in the Volume Table of Contents (VTOC) are overwritten when a data set residing on a volume is deleted. This action is executed during deletion and occurs regardless of the argument specified by ERASE.

The VTOC data set (the physical extent on the volume) can only be overwritten when the entire volume is re-formatted.

Tape Volumes

The reuse of tape volumes is controlled by administrative practices. Specifically, a security administrator is responsible for maintaining a scratch pool of tapes. A scratch pool consists of either new or degaussed tapes.

The administrator may execute the SEARCH command using the EXPIRE operand to determine which volumes are beyond their security retention period. The tape's RACF volume definition is then removed before the tape is manually degaussed and placed back in the scratch pool.

Additionally, a user may relinquish a tape before the retention period is over by notifying the administrator. At this point, the data on the tape has not been deleted and it is the responsibility of the administrator to degauss the tape.

Address Spaces

Data reuse for address spaces is controlled by the translation from a virtual address space to real pages in memory and whether or not that page had been previously referenced. This control is true for all address spaces.

Specifically, a page-fault will occur only when an I/O operation attempts to address a page not currently in real memory. The acquired real page is then overwritten with data from the virtual address space (page-in) or zeros upon this first reference. If a real page is acquired to be used as a virtual page that has never been referenced, the page is overwritten with zeros (an operation of RSM).

Virtual pages are acquired and released through GETMAIN and FREEMAIN routines, respectively. These routines ensure that the page tables for newly acquired virtual pages indicate that the pages have never been referenced. Any attempt to access non-GETMAINED memory, will cause a page-fault to occur and then an ABEND.

An initiator performs further operations in its address space to control data reuse. After the last job step completes, the terminating portion of the initiator performs the following actions: all control blocks associated with the job in LSQA and SQA are released and the ASM is notified to free all of its control blocks for VIO data sets created by the job.

Temporary Data Sets

VIO temporary data sets are controlled as a logical group of pages which reside in an address space. These data sets will remain in the address space for the duration of the job, although the pages may physically reside in auxiliary or real memory. When the job terminates, the pages occupied will be released. This release is controlled by the same paging operations, as explained in the "Address Space" section.

Data reuse for VSAM and non-VSAM temporary data sets is controlled in the same manner as other VSAM or non-VSAM data sets. The only difference is that the temporary data set are overwritten (controlled by the ERASE option) at the termination of the job.

Spool Data Sets

Spool data sets are controlled by JES and are located on a spool volumes specifically allocated for JES. Access to the data in the spool data set is allowed only to the owners and authorized programs. If the spool data sets are destroyed, then the ERASE option causes the data sets be overwritten.

TPUT Messages

The TPUT macro supported by TSO is used to pass TPUT messages to VTAM. Specifically, they are placed in the CSA which is protected by its own storage key and therefore unaddressable by users.

The TSO/E SEND command uses TPUT to transmit messages between users. The TPUT operation allows messages to be displayed when the receiving user is logged on or saved until the receiving user logs in.

For TSO/E release 4, messages can be saved to user broadcast data sets or into the SYS1.BROADCAST data set. The SYS1.BROADCAST data set is accessible only by the system operator and is used for public announcements whereas the user broadcast data sets hold messages for individual users. In either case, the data sets are protected by RACF control and belong to specified users. Therefore when deleted by the user, the erase on scratch option will be invoked.

VTAM Logical Units

VTAM Logical Units (LU) manage session parameters when establishing a LU-LU session. LUs are not directly accessible by users, rather an LU is accessed using the VTAM macros (e.g. SEND and RECEIVE).

System Terminals

The system terminals do not have internal memory and are unable to store more than one page of information on the screen. The information displayed on the terminal's screen is scrolled over when further screens are displayed. The final screen of a session is overwritten with a logon screen by TCAS.

AUDITING

MVS/XA audits all operator actions and RACF has the ability to audit user accesses to resources. RACF generates audit records for all security relevant events and stores these records in data sets owned by SMF. The RACF Report Writer generates tailored audit reports from the data in the SMF data set.

Audit of Operator Actions

The MVS hardcopy log contains a record of operator console message traffic. The characteristics of the hardcopy log are defined during IPL. By default, this log is written to SYSLOG, a JES spool data set, and all operator and system commands, responses, and status displays are written to the log.

System Management Facilities

The master scheduler task attaches SMF during IPL. SMF runs in its own address space which contains SMF control blocks and buffers. SMF routines collect data, provide for user-supplied data collection routines, and record the collected data in a data set. The following components contain SMF data collection routines and exits for user-supplied data collection routines:

- The interpreter (JES)
- The initiator/terminator (MVS)
- The command processor (TSO, MVS, RACF)
- The timer supervisor (MVS)
- All storage managers (MVS)
- The system resource manager (MVS)
- JES2 (in general)
- TSO/E (in general)
- RACF

The components listed above build records as they collect data. Once a complete SMF record is built, the component issues SVC 83 to transfer the records to an SMF buffer. SVC 83 then schedules a service request to write data to an SMF data set, and when necessary initiates a switch to another SMF data set.

There are over 60 types of SMF records that record different categories of data, such as machine data, auxiliary storage data, VSAM data set activity, JES2 data, and RACF data. RACF produces two SMF records: record type 80 and record type 81.

Up to 36 SMF data sets may be allocated. These data sets are RACF protected. The primary data set is the one used for recording. The secondary data sets are used when the primary is full. Each data set is a VSAM data set that resides on a single volume, is cataloged, and not extendable. Each SMF data set must be created before the first IPL that starts SMF recording.

When the current recording data set cannot accommodate any more records, the SMF writer automatically switches recording from the active SMF data set to an empty data set, and passes control to the SMF dump exit. The operator is then notified that the data set needs to be dumped. If all SMF data sets are full, SMF will be unable to record data until an SMF data set is dumped. When this condition occurs, audit records will be lost unless an exit has been installed. If this exit is installed, the system will be halted when the all audit logs are full. This exit is provided as part of RACF and can be found in member RACC2EXT of the SYS1.SAMPLIB system data set.

Auditor

Only the auditor is able to perform RACF audit functions. An auditor is a user with the AUDITOR or group-AUDITOR attribute. The group-AUDITOR attribute allows the same authorities as the AUDITOR attribute except that the authorities only apply to a specific group and its subgroups.

The auditor can specify audit controls by using the RACF SETROPTS command or the "Set Audit Options" ISPF panels. General audit controls direct RACF to log the following: changes to profiles; the activities of users with the SPECIAL, group-SPECIAL, OPERATIONS, or group-OPERATIONS attributes; command violations; and access attempts to resources with a specified security level.

Logging

RACF logs data by writing SMF records to an SMF data set. As mentioned above, RACF produces SMF record types 80 and 81.

Record type 81, RACF Initialization Record, is written at the completion of the initialization of RACF. The information contained in this record includes the name and volume identification of each RACF data base, the data set name and volume identification of the UADS data set, RACF options, and the maximum password interval. Record type 80, RACF Processing Record, is written for all other RACF events. RACF writes one record for each event. Included in the type 80 record are: time and date, event code and qualifier, processor identification, user identification, group name, reason for logging, terminal id, and relocate sections. Relocate sections record the name of the object that caused the security event to take place. RACF logs events that are essential to data security.

RACF always logs:

- the use of the RVARY or the SETROPTS command
- the failure of the RACINIT SVC to verify a user during logon attempts
- the action performed by the console operator to grant access to a resource as part of the failsoft processing performed when RACF is inactive

Additionally, the RACF auditor can direct RACF to log the following events:

- use of RACDEF SVC
- changes to RACF profiles
- RACF commands issued by a SPECIAL or group-SPECIAL user
- RACF command violations
- RACF-related activities of specific users
- accesses to resources allowed because the user has OPERATIONS or group-OPERATIONS attribute
- all or some accesses to specific data sets
- all or some accesses to specific general resources

The owner of resources can specify in the resource profile what type and level of accesses to log or that no logging is to occur. Owner-controlled logging is not directly under the control of the auditor, however, the auditor can expand a resource owner's logging specification by issuing RACF auditor commands. The security auditor cannot change or delete the resource owner's logging specification. There is no auditing of accesses granted by Global Access Checking or by FRACHECK.

Complex Auditing

Each CP within a complex maintains its own SMF data sets. In order to determine the audit trail from the entire complex, the SMF data set from each system may be dumped to a common data set and the report writer then run using the common data set as input, or each SMF data set may be specified in JCL as input to the report writer. The system id of each system is recorded in the SMF logs.

RACF Report Writer

The RACF report writer is used to generate reports from the SMF audit records. A user with the AUDITOR or SPECIAL *attribute can run the report writer.* The RACF report writer reformats SMF records and uses these reformatted records as inputs to the modules that produce the reports. To run the report writer, a user must have the AUDITOR or SPECIAL attribute. The RACF report writer may be run as a batch job or it may be run from a RACF command during a TSO/E session.

The input file to the RACF report writer consists of SMF record types 20, 30, 80, and 81. Type 20 records contain job initiation information including job name and RACF group and user name. Type 30 records, written at the termination of a batch job or step, a TSO/E session, or a started task, contain information such as the job identification, termination status, job step start and end times, and storage protect key. SMF records type 80 and 81 are written to the SMF data set by RACF and are described above.

The RACF report writer operates in three phases. The first phase, command and subcommand processing, invokes the report writer and allows subcommands SELECT, EVENT, LIST, SUMMARY, and END to be entered. The SELECT and EVENT subcommands specify which of the input records the report writer uses to generate the reports. Records may be selected by date, time, user, group, or various other criteria. The events that can be used to select records include TSO/E logon, ADDUSER command, and SETROPTS command. The LIST subcommand formats and prints a listing of each SMF record selected while the SUMMARY subcommand formats and prints a summary listing of the SMF records. END terminates subcommand processing and the first phase of the report writer. During the second phase, record selection, the report writer compares the SMF records against the criteria specified in the SELECT and EVENT subcommands, and reformats the selected records if necessary. During the third phase, report generation, the reports requested with the LIST and SUMMARY subcommands are generated. The report writer produces reports for the LIST subcommand by listing all SMF records from the work data set in the sequence that has been specified. For each SUMMARY subcommand, the report writer produces a separate summary report of the SMF record by group, resource, command, RACF event, or owner activity (depending on what was specified).

Data Security Monitor

In addition to the RACF report writer, the RACF data Security Monitor (DSMON) produces eleven standard reports regarding the status of RACF controls. To run DSMON, a user must have the AUDITOR attribute or must have at least READ authority in the DSMON profile access list. The eleven standard reports produced are:

- System report - contains the identification of the operating system and specifies which version of RACF is installed and whether it is active.

- Group tree report - lists all subgroups for the SYS1 or the user supplied group.

- Program properties table report - lists all the programs in the program properties table, whether each is authorized to bypass password protection, and whether each runs with a system key.

- RACF authorized caller table report - lists the names of all programs in the RACF authorized-caller table and indicates whether each program is authorized to issue the RACINIT- or RACLIST SVC.

- RACF class descriptor table report - lists class name and status for all general resource classes in the class descriptor table.

- RACF exits report - lists the names of all the installation-defined RACF exit routines.

- RACF global access checking table report - lists all entries in the global access checking table.

- RACF started procedures report - lists each entry in the started procedures table.

- Selected user attribute report - lists all RACF users with the SPECIAL, OPERATIONS, AUDITOR, or REVOKE attribute and indicates if they are at the system or group level.
- Selected user attribute report summary - shows total number of installation-defined users and of users with the SPECIAL, OPERATIONS, AUDITOR, and REVOKE attributes at the system and group level.
- Selected data sets reports - lists all data sets that meet one or more of the selection criteria (e.g. APF authorized, RACF backup)

RACF provides auditing of security relevant events by logging the events to the SMF data set. The events to be audited are selected by a user with the AUDITOR or group-AUDITOR attribute. The RACF Report Writer can then be used to generate reports from the SMF audit records.

ASSURANCES

This chapter discusses the assurances provided by the IBM MVS/XA with RACF.

FUNCTIONAL TESTING

Software and hardware development on IBM systems is controlled by engineering cycles. These cycles include design, development, and test phases. This section provides an overview of the testing involved in these phases.

Software Testing

IBM employs a development guide which has a list of objectives to be met before a software product can be shipped to customers. Two major checkpoints are Design Verification Test and Pre-Shipment Test. The test department bases its goals on these two checkpoints. The test department is independent of the design and development departments.

Software engineering is performed during every phase of product design and development, from walkthroughs of the high level design to a complete system test with finished products. To control the testing at each phase of development of a product, a comprehensive test plan is produced for every new software product or new version of a software product. This plan includes definition of test objectives, entrance and exit criteria, responsibilities, test schedules, tools to be used, and dependencies of the product on other products. There are ten phases at which a product is tested:

- high level design,
- low level design,
- unit test,
- driver build,
- function test,
- system test,
- installation walkthrough,

- performance test,
- field test,
- install test.

After all the above phases, IBM, through its early support program, provides the product to a limited number of customers. This gives the final assurance that a product is ready for general release. Descriptions of each phase follow.

The high level design phase consists of developing an English description of the new or changed product. Reviews of the specifications and walkthroughs are utilized to accomplish this phase. In the low level design phase algorithms are described using formal tools or methods, recovery considerations are defined, and data layouts are created. This phase has the goal of producing the code.

During unit test the execution of the smallest program unit is verified. For example, all entry points are invoked, each conditional branch is executed both ways, and defined inputs are checked to see if they give the expected outputs. During the driver build phase unit tested code is collected into functioning subsets of the product called drivers. The product's comprehensive test plan is also produced in this phase.

The function test phase exercises the algorithms and the interfaces at a low level. Sources for tests include specifications, participation in design and code walkthroughs, code listing, pre-existing test cases and documentation drafts. The system test phase has the goal of exposing the whole product to a real, production-like environment. There are four parts to this test including the initial functional evaluation, the basic test of the product, the load and stress test of the product, and the characteristic evaluation of the product.

During the installation walkthrough phase the installability of a product is examined. Representatives from development organizations participate in typical installations and migrations of the product. The goal of the performance test is to obtain an estimate on the capacity and the throughput of the product.

The field test phase involves running a pre-install test version of the operating system with all the necessary products and hardware at an internal IBM site. The purpose of this phase is to observe the product in a production environment. The IBM Software Distribution (ISD) sends products to customers. The install test phase makes sure that the materials the IBM Software Distribution receives are a complete set.

Hardware Testing

Hardware functional testing is an important part in the total test scheme for a modem computer product. Rather than assume the hardware works by default (i.e., the software runs), a set of tests designed to ensure that the architecture implemented by the machine conforms to the specification given in the *Principles of Operation* is run on every machine that leaves the manufacturing facility. These tests have the additional advantage of being able to be run after the system is in place to ensure correct functioning of the hardware, and also to aid in hardware failure detection.

The system that IBM has devised to perform this function for the System 370-XA architecture is the Systems Assurance Kernel (SAK). This subsystem consists of approximately one million lines of code, and can be used to test other architectures such as S/370 and 370-XA. This subsystem is used at all points in the manufacturing life-cycle of a hardware product, including sub-system simulation, element simulation, system simulation, assurance verification testing and manufacture verification testing, final verification testing and field verification and maintenance testing.

The SAK can support a number of environments, such as multiprogramming, multiprocessing, extended addressing, and V=V and V=R modes. The test programs are based on a series of random instructions or command sequences with random data patterns where ever possible. Testing covers all aspects of the final system, including the central processor, cache, main and extended memory, and the storage controller, and portions of the channel subsystem such as the channels themselves, the DASD and tape device operations, CTC functioning, and other device operation (e.g., printers, card readers).

Failing instructions or operations are logged, and can be isolated so that correction of the error is facilitated. A number of new test generation techniques, as well as testing methodologies, were devised and implemented in the SAK. Typical SAK operation involves hosting the majority of the SAK software on a separate machine, compiling the test case, and then downloading it to the target machine. After the run, all data collected from the run is analyzed on the host. SAK is run on all machines before leaving the factory to validate their conformance to the 370-XA architecture. Field engineers also have a scaled-down version of SAK which they can use to suppliment the diagnostics that are resident on the machines which they service.

MAINTENANCE and INSTALLATION

System Modification Program

IBM provides a Program Update Tape (PUT) usually distributed every six weeks which contain updates and revisions to MVS/XA. The revisions are referred to as Program temporary fixes (PTF).

For an installation to handle and manage PTFs effectively, IBM furnishes the System Modification Program (SMP). This program provides the system administrator with a means to add or change the system or its parameters. In addition, SMP/E provides a means to update library functions, revise system modules, define macros not available during the initial product load, and is used during system generation.

SMP executes authorized but since it is used only by system programmers during maintenance and system generation (before the system is operating in a trusted mode), it is not part of the evaluated software.

System Generation

The purpose of a system generation is to build the required MVS libraries that suit the needs of an installation. Through this sysgen process, modules are selectively chosen from the distributed library and then placed into the actual system library. In addition, the sysgen will create the necessary I/O blocks required by the selected system.

Presently, IBM offers four methods to generate an MVS system for its customers. The customer is responsible for choosing one of these methods. They are direct product order, Custom Build Product Delivery Option (CBPDO), Custom Build Installation Productivity Option (CBIPO), and MVS express.

The direct product order method delivers separate tapes and manuals for each product. A product program directory provides the necessary information to cover installation. SMP/E plays a vital role for this method of installation because each product must be loaded into its data base.

The CBPDO helps to automate the direct method by delivering one logical tape and to standardize the licensing agreement for all of the products ordered. The customer is responsible for selecting all PTF to be included the installation of the tape using SMP/E.

The CBIPO option delivers a full system which has had the software configuration checked. It also automates the ordering and licensing process, similar to CBPDO. One package is delivered containing all software and documentation required to install the system. The documentation is centered around a model installation, to help assist the customer. The customer is responsible for loading the tape to a disk and selecting the system configuration.

MVS express is a service which is intended for first time customers who have been recommended by their IBM representative. This service minimizes the customer's effort by selecting system components and actually IPLing the system.

DIAGNOSTICS

IBM provides several utility programs that can be used to verify the correct operation of the hardware and firmware.

On-Line Test Executive Program (OLTEP)

OLTEP is an IBM utility for executing the online programs that test all evaluated control units and devices. OLTEP is a standard component of the operating syste .n, and resides in the system libraries. It runs as a system job, causing minimum interference with the normal system operations (i.e., other jobs can be run while OLTEP is running). This program performs diagnostics, prints the diagnostic information, and verifies repairs for control units and I/O devices.

SYS1.LOGREC Error Recording Data Set

The system error recording programs (ERP) write information about all hardware failures, selected software errors, and system conditions into the SYS1.LOGREC data set. These programs consist of the Recovery Termination Manager and the Error Recovery Programs (one for each device type). The records in this data set can contain either error statistics or environmental data. Examples of error statistics are channel or I/O device count failures, times of system failure and hardware status at time of failure. Examples of environmental data include time and circumstances for each failure, and device/control unit and software system recovery attempt results. These records are recorded in chronological order.

The types of events recorded in SYS1.LOGREC include operator initiated system termination, serious errors which result in abnormal termination (operator intervention required), system initialization (IPL), buffer overflow, paging I/O errors (for permanent channels), and I/O device failures (both temporary and permanent).

Processor Complex Exerciser (PCX)

PCX is a system exerciser program for the IBM 3090 Processor Complex. It tests the following functions: the dynamic address translation mechanism, the address space instructions (i.e., program calls, program transfers, set primary ASN, load address space parameters), the general processor instruction set, the multi-processing instructions, contention between processors, storage protection, main storage and the processor buffers, the page in/page out instructions, and the I/O channel subsystem.

Built-In Diagnostics

Both the 3090 and 4381 have extensive diagnostics built in which can be run on demand to ensure the correct operation of the hardware, and help isolate problems in the machines. These diagnostics are available at the MVS operator's console on the 4381, and on the system and service (level 2) consoles at the PCE for the 3090. Diagnostics which are activated at the time of fault can actually detail to the field engineer the part which is in error on the machine, and has proven to be extremely reliable thus far. A facility exists that will call an IBM service center with a problem report at the time of failure. This option is site-configurable, and requires operator confirmation before it proceeds.

CONFIGURATION MANAGEMENT

IBM maintains a comprehensive hardware product lifecycle. The hardware is designed, prototypes developed, and comprehensive test plans developed before the hardware product is manufactured. Once the hardware has been installed, they have strict guidelines for the control and evaluation of engineering changes. One committee acts as a single interface for all changes made to a particular product. This committee is also responsible for ensuring the changes are initiated and accomplished, including the evaluation, monitoring, and reporting of all changes. IBM also has strict guidelines for the software lifecycle including initial specifications, final specifications, comprehensive test plan, design reviews, design verification testing, and code inspection, all prior to integration. The security strategy is defined in the program specification stage.

Once software has been distributed, it may need corrections based on an Authorized Program Analysis Report (APAR). An APAR is the description of a program, documentation, or program distribution error on a currently supported program release. APARs can be written and submitted to IBM by specified users of the programs. The organization originating a change coordinates the change with all other organizations which may be impacted. Modifications are made to both software and documentation. New changes to software are tested and regressions tests are performed. All changes are kept track of using an on-line configuration management tool.

EVALUATION AS A C2 SYSTEM

Discretionary Access Control

Requirement

> The TCB shall define and control access between named users and named objects (e.g., files and programs) in the ADP system. The enforcement mechanism (e.g., self/group/public controls, access control lists) shall allow users to specify and control sharing of those objects by named individuals, or defined groups of individuals, or by both, and shall provide controls to limit propagation of access rights. The discretionary access control mechanism shall, either by explicit user action or by default, provide that objects are protected from unauthorized access. These access controls shall be capable of including or excluding access to the granularity of a single user. Access permission to an object by users not already possessing access permission shall only be assigned by authorized users.

Applicable Features

The objects in MVS/XA with RACF are DASD data sets, spool data sets, temporary data sets, tape volumes, address spaces, volume tables of contents, TPUT messages, catalogs, VTAM logical units, and terminals. Subjects in MVS/XA with RACF are address spaces performing user and system functions. There are five types of subjects: console operators, started tasks, system services, TSO/E users, and batch jobs page 93, "Mapping Subjects to Userids".

Users are permitted to share DASD data sets and tape volumes. In addition, a site may control access to terminals. MVS/XA with RACF uses RACF profiles to list the userids having access to DASD data sets, tape volumes, and terminals.

Spool data sets, temporary data sets and address spaces are not shared. These objects are controlled such that only the object's owner may access the object.

The remaining objects are system objects. These objects are manipulated by the system on behalf of a subject. A subject's ability to read or modify one of its objects is based upon the subject's access to another object. For example, a TSO/E user's address space may only read from the VTAM logical unit associated with the user's terminal. For more information on the access controls used to determine a user's access to an object see page 96, "DISCRETIONARY ACCESS CONTROL".

The contents of a RACF profile (see page 83, "Resource Profiles") identifies the individuals and groups having access to the protected object, along with an access authority (i.e., NONE, READ, UPDATE, CONTROL and ALTER). A profile may be modified by users having ALTER access to an object, and by users owning the profile.

The RACF option PROTECT=ALL, allows a data set to be created only if it will be protected by a RACF p^r^ofile. That is, a data set must either have a discrete profile or be protected by a generic profile.

Tape volumes are defined to RACF by creating a profile for the tape volume. Until a tape volume has been defined the tape volume is protected by data management routines (e.g., DFP) using passwords.

When additional terminals are added to the system, they are protected immediately if the RACF TERMINAL class is active. If the TERMINAL class is not active then no terminals are protected.

A RACF profile can be used to specify the access individual users have to DASD data sets, tape volumes and terminals. The access authority of ALTER is required to modify the contents of the profile, thereby granting access permission to other users.

Conclusion

MVS/XA with RACF satisfies the C2 Discretionary Access Control requirement.

Object Reuse

Requirement

> All authorizations to the information contained within a storage object shall be revoked prior to initial assignment, allocation, or reallocation to a subject from the TCB's pool of unused storage objects. No information, including encrypted representations of information, produced by a prior subject's actions is to be available to any subject that obtains access to an object that has been released back to the system.

Applicable Features

MVS/XA with RACF uses the RACF ERASE option, administrative practice, and other software to address the object reuse requirement.

The RACF ERASE option, when set to ALL, ERASE(ALL) causes all DASD data sets to be overwritten with zeros at the time of deletion (release). The deletion is executed by the DFP DELETE function which is initiated by JCL instructions or by the DELETE command.

Spool data sets are controlled by JES and access is allowed only to owners and authorized programs.

VTOC entries are overwritten at the time of deletion. The physical area occupied by the VTOC, referred to as the VTOC data set, can only be overwritten if the entire volume is re-formatted.

Tapes volumes are handled in a procedural manner. Operators select or are instructed to degauss a tape before they are placed in the tape pool. Once properly degaussed, information cannot be obtained from the tape volume.

Address spaces must obtain real page before data can be read or written. A real pages is overwritten with user data or zeros upon the first reference to the page.

VIO temporary data sets reside within an address and are subject to the paging operations described above. VSAM and non-VSAM temporary data sets are controlled by the ERASE option.

TPUT messages and VTAM LU place information in the VTAM address space which is key protected and inaccessible by users. If the address space is terminated, space is released as described in address space section.

The system terminals do not store information. Information to the terminal is functionally overwritten as further screens are displayed. Furthermore, a LOGON screen overwrites any remaining information after the user logs off.

Conclusion

MVS/XA with RACF satisfies the C2 Object Reuse requirement.

Identification and Authentication

Requirement

> The TCB shall require users to identify themselves to it before beginning to perform any other actions that the TCB is expected to mediate. Furthermore, the TCB shall use a protected mechanism (e.g., passwords) to authenticate the user's identity. The TCB shall protect authentication data so that it cannot be accessed by any unauthorized user. The TCB shall be able to enforce individual accountability by

> providing the capability to uniquely identify each individual ADP system user. The TCB shall also provide the capability of associating this identity with all auditable actions taken by that individual.

Applicable Features

MVS/XA with RACF requires all users except console operators to identify themselves before they can request any other action to be performed. This identification is accomplished via userids. Each user has a unique userid.

MVS/XA with RACF uses passwords to authenticate users. At logon each interactive user is required to supply a userid and password. For batch jobs, either the userid and password are supplied by the submitter, or the current validated userid and password are propagated along with the batch job. Started tasks have a userid and password defined in the userid/groupid replacement table, which is checked before the task can execute.

Password data are stored in a hashed, masked, or encrypted form in the RACF data base. Only authorized users can access the RACF data base

The MVS/XA with RACF TCB maintains and protects userids through all the steps of job/task execution. Each userid can thus be used to associate all auditable actions with the subject.

Conclusion

MVS/XA with RACF satisfies the C2 Identification and Authentication requirement.

Audit

Requirement

> The TCB shall be able to create, maintain, and protect from modification or unauthorized access or destruction an audit trail of accesses to the objects it protects. The audit data shall be protected by the TCB so that read access to it is limited to those who are authorized for audit data. The TCB shall be able to record the following types of events: use of identification and authentication mechanisms, introduction of objects into a user's address space (e.g., file open, program initiation), deletion of objects, actions taken by computer operators and system administrators and/or system security officers, and other security relevant events. For each recorded event, the audit record shall identify: date and time of the event, user, type of event, and success or failure of the event. For identification/authentication events the origin

> of request (e.g., terminal ID) shall be included in the audit record. For events that introduce an object into a user's address space and for object deletion events the audit record shall include the name of the object. The ADP system administrator shall be able to selectively audit the actions of any one or more users based on individual identity.

Applicable Features

RACF has the capability to audit all security relevant events by recording the events in an SMF data set. To prevent the loss of audit data when the SMF data set is full, an installation has the option, through the use of an exit, to halt the system when this condition occurs. The SMF data set is protected by RACF. Only users with the AUDITOR or SPECIAL attribute may generate reports from the SMF audit records.

The owner of each resource and the system auditor specify which events are to be recorded in the SMF data set. Types of events that may be audited include use of the RACINIT SVC, accesses to data sets and general resources, RACF command violations, and RACF-related activities of specific users. The information recorded in the SMF record includes, userid, groupid, event code, date and time of event, success or failure, terminal ID, and object name (if applicable).

The user with the AUDITOR or SPECIAL attribute may use the RACF Report Writer to generate audit reports from the records in the SMF data set. The SELECT subcommand of the report writer provides the means by which the AUDITOR may select individual user actions to appear in the audit report. The EVENT subcommand provides the means to select one particular event to be included in the audit report. For a full discussion of Audit see page 107, "AUDITING".

Conclusion

MVS/XA with RACF satisfies the C2 Audit requirement.

System Architecture

Requirement

> The TCB shall maintain a domain for its own execution that protects it from external interference or tampering (e.g., by modification of its code or data structures). Resources controlled by the TCB may be a defined subset of the subjects and objects in the ADP system. The TCB shall isolate the resources to be protected so that they are subject to the access control and auditing requirements.

Applicable Features

The System 370-XA architecture provides many mechanisms which are used by MVS/XA with RACF to preserve the integrity of the TCB. Address spaces are isolated from one another by the hardware, using the address space translation mechanism. All TCB components reside in address spaces separate from user address spaces, and are protected from modification by means of the Authorized Program Facility, the two state architecture, and key-controlled memory protection.

All subjects and objects defined in the system are controlled by the TCB. These subjects and objects have been clearly identified and are governed by the access control policy implemented by the system. In addition, extensive auditing can be performed on these subjects and objects.

Conclusion

MVS/XA with RACF satisfies the C2 System Architecture requirement.

System Integrity

Requirement

> Hardware and/or software features shall be provided that can be used to periodically validate the correct operation of the on-site hardware and firmware elements of the TCB.

Applicable Features

MVS/XA with RACF provides many different mechanisms for assuring the on-site hardware listed in Appendix A. The On-Line Test Executive Program is used to extensively test all processors, channels, and control units in the hardware configuration. It also can test devices such as DASD and tape drives for correct operation. The error recording functions included with MVS/XA itself allow for recording in the SYS1.LOGREC error recording data set, and can log various statistics regarding the performance and current availability of hardware components of the machine. A processor complex exerciser is also provided for use with multi-CP 3090 machines. This exerciser will test each CP individually for critical functions, including address space manipulations and I/O subsystem operation.

Conclusion

MVS/XA with RACF satisfies the C2 System Integrity requirement.

Security Testing

Requirement

> The security mechanisms of the ADP system shall be tested and found to work as claimed in the system documentation. Testing shall be done to assure that there are no obvious ways for an unauthorized user to bypass or otherwise defeat the security protection mechanisms of the TCB. Testing shall also include a search for obvious flaws that would allow violation of resource isolation, or that would permit unauthorized access to the audit or authentication data.

Applicable Features

The evaluation team performed testing of the security features of MVS/XA with RACF in April and May, 1988 at an IBM site. The testing was performed in several stages, involving different hardware configurations. The following hardware was used in testing:

3090-400 processor
3090-120 processor
3480 tape drives
3420 tape drives
3274 terminal controller
3278 terminals
3279 terminals
3179 terminals
3880 DASD controller
3380 DASD
3480 tape controller
3422 tape controller
3800-3 printer
4248 printer

The evaluation team first generated the operating system to be tested using IBM's CBIPO system generation process. The team chose the CBIPO method because it delivered the system in one package.

The generated system was then IPL'd on a partitioned 3090-400 system. The C2 system was configured using guidance provided in the RACF System Administrators Guide. The team went on to execute approximately 70 percent of the vendor's security test cases. The test cases focused on the correct functionality of the security mechanisms. Each test case contained a description of the test, one or more variations of the test, instructions, and set-up requirements. The test cases exercised RACF options, PROTECTALL (DAC), ERASE(ALL) (Object Reuse), and BATCHALLRACF (Identification and Authentication); the use of RACF attributes and resource access authorities (DAC); TSO/E job submission (Identification and Authentication); and auditing of RACF commands.

The vendor tests were augmented with tests developed by the team. These tests included filling up the audit log and observing the results, checking the integrity of the backup RACF database, trying to read and/or write beyond a data set extent, and verifying that the FRACHECK macro is never called.

Testing was also performed on a whole 3090-400 (no partitioning)and on a 3090-120. The team verified that a system could be added to a secure complex without compromising security. The team also produced an audit report to verify that the order of a job's phases could be determined in a complex even if the systems' clocks were not synchronized.

Testing revealed the system to work as claimed with the exception of one flaw found in the Report Writer. This tool, used in generating audit reports, was unable to extract logoff audit records from the audit log. IBM corrected this deficiency and the team retested the system proving that the flaw was corrected.

Security Features User's Guide

Requirement

> A single summary, chapter, or manual in user documentation shall describe the protection mechanisms provided by the TCB, guidelines on their use, and how they interact with one another. Evaluation as a C2 system

Applicable Features

There are several documents which collectively provide users with guidance on the security features offered by the evaluated system.

TSO Extensions User's Guide (SC28-1333) and *TSO Extensions Command Language Reference* (SC28-1307) manuals present information about the interactive interface to MVS/XA and the allowable user commands. These manuals describe the logon and logoff procedures and general data manipulation.

*RACF General Information Manual (*GC28-0722) provides summary information about RACF. *RACF User's Guide* (SC28-1341) and the associated Technical Newsletter (TNL SN28-1218) describe the protection mechanisms, their interactions, and functions. The manual, *RACF CommandLanguage Reference* (SC28-0733), defines the syntax and functions of RACF commands supplying information on password manipulation and access controls. The manuals are supplemented with examples of most features previously described. References to other applicable manuals are also included.

Conclusion

MVS/XA with RACF satisfies the C2 Security Features User's Guide requirement.

Trusted Facility Manual

Requirement

> A manual addressed to the ADP system administrator shall present cautions about functions and privileges that should be controlled when running a secure facility. The procedures for examining and maintaining the audit files as well as the detailed audit record structure for each type of audit event shall be given.

Applicable Features

There are several documents which collectively provide system administrators with information describing various security features.

RACF General Information Manual (GC28-0722) provides summary information about RACF. *RACF Security Administrator's Guide* (SC28-1340) and the associated Technical Newsletter (TNL SN28-1217) contain information about planning and establishing a secure system. It is intended for security administrators, group administrators, and other administrators responsible for the MVS/XA system security. The Guide provides an overview of RACF and various administrative information in addition to information and examples of: defining and deleting RACF groups, users, and resources; selecting RACF options; using certain installation procedures; tape handling procedures and essential security features necessary to operate an MVS/XA system at the C2 level. This guide also contains references to other IBM publications for more detailed discussions of related products and features.

RACF Auditor's Guide (SC28-1342) is intended for those individuals defined as RACF auditors. This guide includes such information as an overview of the role of the RACF auditor, a description of the use of the auditing tools, a description of the RACF report writer, and a summary of the Data Security Monitor functions. *RACF: System Programming Library* (SC28-1343) provides a detailed description and format of the audit records.

Finally, *MVS Security* (GC28-1400) is intended to provide the security administrators with an overview information about the MVS products and features that are available to support the implementation of a security program. In particular, the RACF product is presented with examples of how it can be used to protect the system's resources.

Conclusion

MVS/XA with RACF satisfies the C2 Trusted Facility Manual requirement.

Test Documentation

Requirement

> The system developer shall provide to the evaluators a document that describes the test plan, test procedures that show how the security mechanisms were tested, and results of the security mechanisms' functional testing.

Applicable Features

IBM's objective for testing is to insure that a product behaves as defined in the initial specifications. This is achieved through various tests at each stage of development of a product. At every stage of development there are four goals of testing: elimination of defects, exercise of usable functions, reduction of maintenance costs, and achievement of RAS (Reliability, Availability, Serviceability).

IBM has a comprehensive test plan produced for every product on the system. This test plan includes suites of test cases for each product in the system. Each test case contains a number of variations, a description of that test case, and setup/execution instructions.

Due to the size of the TCB, inclusion of specific tests of every variation was not done. The vendor identified the security relevant tests from the available test suites, and testing concentrated on validating a sample of all the security mechanisms.

Conclusion

MVS/XA with RACF satisfies the C2 Test Documentation requirement.

Design Documentation

Requirement

> Documentation shall be available that provides a description of the manufacturer's phiiosophy of protection and an explanation of how this philosophy is translated into the TCB. If the TCB is composed of distinct modules, the interfaces between these modules shall be described.

Applicable Features

IBM has developed a very extensive set of publications describing the philosophy, architecture and design logic of MVS/XA and other evaluated products. At the highest level, *System 370 Extended Architecture Principles of Operation* (SA22-7085) spells out the underlying architecture. MVS/XA, described in *MVS/XA General Information Manual* (GC28-1118) and in a voluminous set of *MVS/XA System Logic Library* (LY28-nnnn) manuals, adheres to this architecture. Likewise, general information manuals and logic manuals for DFP, JES2, ACF/VTAM, TSO/E, and RACF provide a corresponding level of detail.

MVS/XA Overview (GC28-1348) discusses the system and its components at an introductory level while the logic manuals detail the functions before proceeding with detailed flow diagrams and data structure definitions. *MVS Security* (GC28-1400) describes the interfaces and interactions between RACF and other TCB components.

Conclusion

MVS/XA with RACF satisfies the C2 Design Documentation requirement.

This page intentionally left blank.

EVALUATOR'S COMMENTS

System Characteristics

From the security point of view, RACF allows its users a very extensive set of discretionary controls over many system resources. RACF also provides for a role separation between the system administrator, the auditor, and the operator. The auditor may elect to perform selective audits based on several criteria.

User interactions with the system are significantly simplified with ISPF, a tool which provides interactive users with a menu-driven interface to TSO, RACF, and other utilities. ISPF does a good job of partially masking the archaic batch and file systems. At the programmer level, a robust interface to MVS is evident.

User Passwords

The system is capable of either encrypting or masking user passwords. Since password masking alone is perceived as weak, the team recommends that installations utilize the encryption mechanism as the default.

System Documentation

The team has mixed feelings about the detail contained in various system manuals. At the low level, program logic manuals (PLMs) are full of information including flow diagrams, calling sequences, data structures, etc. However, at the end-user level the documentation is often repetitious, inconsistent, and lacking in detail. There are referrals to other manuals which do not add anything new to the information found elsewhere. This situation is particularly obvious in three products: JES, RACF, and VTAM. Finally, the terminology used is frequently inconsistent and confusing.

Testing

The team has mixed feelings about the overall testing approach. IBM implements extensive testing guidelines which are applied to all of the evaluated products. These guidelines result in a thorough coverage of system functions. However, primarily due to logistics various products are never tested together. As a result they all utilize tailored environments which often make them incompatible with one another. The team feels that assembling all these tests together would greatly enhance the entire product testing and aid the evaluation. Supplying results analysis tool would also be welcome.

On the more positive side, the team is impressed with IBM's responsiveness in providing a good training and testing environment, and in addressing all the problems which surfaced during the testing phase of the evaluation.

Closing Comments

While this evaluation, like many others, had its difficulties, the team is looking forward to possible future endeavors with IBM. We also hope that some evaluation knowledge has been conveyed.

EVALUATED HARDWARE COMPONENTS

Processors

3090 Models

Single Processors: 120,120E,150,150E,180,180E
Multiple Processors: 200,280E,300E,400,400E,500E,600,600E

4381 Models

Single Processors: 11 12 13 21 22 23
Dual Processors: 14 24

DASD

Controllers:

3880 Models 3, 21 (for paging only), 23

3990 Models 1, 2, 3

Devices:

3380 Model A4 (Head of string)
Model AA4 (Head of string; need this one for 3880 M x13)
Models AD4, AE4, AJ4, AK4
Model B4
Note: B4 3380 needs an A4 or AA4 3380
Models BD4, BE4, BJ4, BK4, CJ2

3375 Model A1 (head of string) Model B1 Note: B1 3375 needs an A1
3375 Model D1 Note: D1 3375 needs an A1 3375

3350 Models A2, B2 (Note: both need 3880-21 controller)

Tapes

Controllers:

3480 Models A11, A22

Devices:

3422 Models A01, B01
3480 Models B11, B22

Terminals

Controllers:

3174 Model 1L
3274 Models 21A, 31A, 21B, 21D, 31D, 41A, 41D

Devices:

3178 Model 1
3179 Models 1, 2, G1, G2
3180 Models 1, 2
3278 Models 2A, 3, 4, 5
3279 Models 2A, 2B, 3A, 3B

Printers

Controllers and Devices:

3800 Models 3, 6

Devices:

3820 Model 1
4245 Models D12, D20
4248 Model 1
3262 Models 3, 5, 13

EVALUATED SOFTWARE COMPONENTS

TCB Software

MVS/SP JES2	Version 2,	Release 2
MVS/XA DFP	Version 2,	Release 3
ACF/VTAM	Version 3,	Release 1.1 for XA
TSO/E	Version 1,	Release 4 for XA
RACF	Version 1,	Release 8

The system was at a PUT level 8801.

RACF maintenance installed had the following PTF numbers:
UY13757 UY13759 UY13761 UY13763 UY13951 UY14166
UY14176 UY14306 UY14307 UY14315 UY14468 UY14609
UY15082 UY15125 UY15308 UY15310 UY15493 UY15650
UY15677 UY15755 UY15757 UY15759 UY15911 UY15913
UY15924 UY16227 UY16444 UY16445 UY16460 UY16461
UY16674 UY16676 UY16826 UY17174 UY17479 UY17481
UY17552 UY17555 UY17786 UY17804 UY17805 UY17854
UY17944 UY17945 UY18010 UY18098 UY18197 UY18234
UY18235 UY18317 UY18321 UY18325 UY18326 UY18360
UY18416 UY18555 UY18556 UY18574 UY18653 UY18668
UY18698 UY18854 UY19012 UY20723.

The APAR numbers for the RACFRW and RACC2EXT were OY14487 and OY15090, respectively.

Software Outside the TCB

Software outside the TCB, that may be added to the system without affecting the rating, must have the following characteristics:

- can not run in supervisor state,
- can not run APF authorized,
- can not run with key 0 through 7.

This page intentionally left blank.

ACRONYMS

ACB	Access method Control Block
ACEE	Accessor Environment Element
ACF/VTAM	Advanced Communications Function/Virtual Telecommunications Access Method
ADP	Automatic Data Processing
ADSP	Automatic Data Set Protection
ANSI	American National Standard Institute
APAR	Authorized Program Analysis Report
API	Application Program Interface
APF	Authorized Program Facility
ASCB	Address Space Control Block
ASCII	American Standard Code for Information Interchange
ASM	Auxiliary Storage Manager
ASN	Address Spacc Number
ASR	Address Space Register
ASXB	Address Space Control Block Extension
AUK	Authorized User Key
BAM	Block Availability Mask
BCD	Binary Coded Decimal BDAM Basic Direct Access Method
BISAM	Basic Indexed Sequential Access Method
BPAM	Basic Partitioned Access Method
BSAM	Basic Sequential Access Method
CBPDO	Customer Built Product Delivery Option
CBIPO	Customer Built Installation Productivity Option
CCE	Channel Control Elements
CP	Central Processor
CPU	Central Processing Unit
CSA	Common Save Area
CTC	Channel to Channel adaptor
CTSS	Compatible Time-Sharing System
DAC	Discretionary Access Control
DASD	Direct Access Storage Device
DAT	Dynamic Address Translation
DCB	Data Control Block
DEB	Data Extent Block
DFP	Data Facilities Product

DoD	Department of Defense
DSCB	Data Set Control Block
DSMON	Data Security MONitor
EBCDIC	Extended Binary-Coded-Decimal Interchange Code
ECB	Event Control Block
ECSA	Extended Common Storage Area
ELSQA	Extended Local System Queue Area
EPL	Evaluated Products List
ERP	Error Recording Program
ESA	Expanded Storage Arrays
EXCP	Execute Channel Program
FLIH	First Level Interrupt Handler
FLPA	Fixed Link Pack Area
GB	Gigabyte
GRS	Global Resource Serialization
GSPL	Global Service Priority List
HASP	Houston Automatic spooling Priority
IBM	International Business Machines
ICB	Index Control Block
IML	Initial Microprogram Load
IMS	Information Management System
IOB	Input/Output Block
IOS	Input/Output Supervisor
IOSB	Input/Output Supervisor Block
IPL	Initial Program Load
IRIM	IPL Resource Initialization module
ISO	International Standards Organization
ISPF	Interactive System Productivity Facility
JCL	Job Control Language
JES	Job Entry Subsystem
JES2	Job Entry Subsystem 2
JES3	Job Entry Subsystem 3
LLA	LNKLST Lookaside
LRU	Least Recently Used
LU	Logical Unit
LPA	Link Pack Area
LRU	Least-Recently Used
LSA	Logic Support Adaptors
LSMQ	Local Service Management Queue

LSPL	Local Service Priority List
LSQA	Local System Queue Area
LSS	Logic Support Stations
MB	Megabyte
MFT	Multiprogramming with a Fixed number of Tasks
MLPA	Modified Link Pack Area
MVT	Multiprogramming with a Variable number of Tasks
MVS	Multiple Virtual Storage
NAU	Network Addressable Units
NCSC	National Computer Security Center
NIB	Node Initialization Block
OLTEP	On-Line Test Executive Program
OS	Operating System
PCE	Processor Control Element
PCP	Primary Control Program
PCX	Processor Complex Extension
PDS	Partitioned Data Set
pel	picture elements
PID	Program Information Division
PLPA	Pageable Link Pack Area
PMA	Processor Memory Array
PP	Physically Partitioned
PSA	Prefix Save Area
PSW	Processor Status Word
PT	Program Transfer
PTF	Program Temporary Fix
PU	Physical Unit
PUT	Program Update Tape
QISAM	Queued Indexed Sequential Access Method
QSAM	Queued Sequential Access Method
RACF	Resource Access Control Facility
RAS	Reliability Availability Serviceability
RBA	Relative Byte Address
RCT	Region Control Task
RIM	Resource Initialization Module
RPL	
RSM	Real Storage Manager
SAK	System Assurance Kernel
SAT	System Authorization Table
SCE	System Control Element

SCII	Standard Code for Information Interchange
SI	Single Image
SLT	System Linkage Table
SMF	System Management Facility
SMP	System Modification Program
SQA	System Queue Area
SRB	Service Request Block
SRM	System Resource Manager
SSCP	System Service Control Point
STC	Started Task Control
SVC	SuperVisor Call
SWA	Scheduler Work Area
TCAM	Terminal Communications Access Method
TCAS	Terminal Communication Address Space
TCB	Trusted Computing Base
TCB	Task Control Block
TCSEC	Trusted Computer Security Evaluation Criteria
TLB	Translation Lookaside Buffer
TMP	Terminal Monitor Program
TOD	Time Of Day
TSO/E	Time Sharing Option/Extensions
TVTOC	Tape Volume Table Of Contents
UACC	Universal Access Authority
UADS	User Attributes Data Set
UCB	Unit Control Block
VIO	Virtual Input/Output
VM	Virtual Machine
VS	Virtual Storage
VSAM	Virtual Storage Access Method
VSM	Virtual Storage Manager
VSPC	Virtual Storage Personal Computing
VTAM	Virtual Telecommunications Access Method
VTIOC	Virtual Telecommunications I/O Coordinator
VTOC	Volume Table Of Contents
XA	Extended Architecture

☆U.S.GOVERNMENT PRINTING OFFICE:1989-622-730/10100

UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE

REPORT DOCUMENTATION PAGE

1a. REPORT SECURITY CLASSIFICATION
UNCLASSIFIED

1b. RESTRICTIVE MARKINGS
None

2a. SECURITY CLASSIFICATION AUTHORITY

2b. DECLASSIFICATION/DOWNGRADING SCHEDULE

3. DISTRIBUTION/AVAILABILITY OF REPORT
Approved for public release;
Distribution Unlimited

4. PERFORMING ORGANIZATION REPORT NUMBER(S)
CSC-EPL-88/003

5. MONITORING ORGANIZATION REPORT NUMBER(S)
~~S230,621~~ → S 230620

6a. NAME OF PERFORMING ORGANIZATION
National Computer Security Center

6b. OFFICE SYMBOL (If applicable) C12

7a. NAME OF MONITORING ORGANIZATION

6c. ADDRESS (City, State and ZIP Code)
9800 Savage Road
Ft. George G. Meade, MD 20755-6000

7b. ADDRESS (City, State and ZIP Code)

8a. NAME OF FUNDING/SPONSORING ORGANIZATION

8b. OFFICE SYMBOL (If applicable)

9. PROCUREMENT INSTRUMENT IDENTIFICATION NUMBER

8c. ADDRESS (City, State and ZIP Code)

10. SOURCE OF FUNDING NOS

PROGRAM ELEMENT NO	PROJECT NO	TASK NO	WORK UNIT NO.

11. TITLE (Include Security Classification)
(U) Final Evaluation Report, IBM MVS/XA with RACF

12. PERSONAL AUTHOR(S)
Stigdon, Dana Nell; Rub, Jerzy*; Elliott, Ken*; Glabus, Cindy*; Oehler, Michael; Haley, Cornelius (* Aerospace Corp.)

13a. TYPE OF REPORT
Final

13b. TIME COVERED
FROM ____ TO ____

14. DATE OF REPORT (Yr, Mo., Day)
880615

15. PAGE COUNT
152

16. SUPPLEMENTARY NOTATION

17. COSATI CODES

FIELD	GROUP	SUB GR

18. SUBJECT TERMS (Continue on reverse if necessary and identify by block number)
MVS/XA Resource Access Control Facility
Trusted Computer System Evaluation Criteria
EPL C2 NCSC

19. ABSTRACT (Continue on reverse side if necessary and identify by block number)

The security protection provided by International Business Machine Corporation's MVS/XA with RACF was evaluated by the National Computer Security Center (NCSC). The NCSC evaluation team has determined that the highest class at which MVS/XA with RACF satisfies all the specified requirements of the Trusted Computer System Evaluation Criteria, dated December 1985, is class C2, and therefore has been assigned a class C2 rating by the NCSC.

A system that has been evaluated as being a class C2 system provides a Trusted Computing Base (TCB) that enforces discretionary access control and, through the use of audit capabilities, accountibility for the actions users initiate.
This report documents the findings of the evaluation.

20. DISTRIBUTION/AVAILABILITY OF ABSTRACT
UNCLASSIFIED/UNLIMITED

21. ABSTRACT SECURITY CLASSIFICATION
UNCLASSIFIED

22a. NAME OF RESPONSIBLE INDIVIDUAL
DENNIS E. SIRBAUGH

22b. TELEPHONE NUMBER (Include Area Code)
(301)859-4458

8b. OFFICE SYMBOL
C/C12

EDITION OF 1 JAN 73 IS OBSOLETE.

UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE